AD-A055 581 DAVID W TAYLOR NAVAL SHIP RESEARCH AND DEVELOPMENT CE--ETC F/G 9/2
THE SHARP DATA BASE MANAGEMENT SYSTEM - USER INFORMATION MANUAL--ETC(U)
AUG 76 B WALLIS, R HYDE, T BLAKE

UNCLASSIFIED DTNSRDC-TM-188-76-1 NL

1 OF 3
AD
A055 581

DAVID W. TAYLOR NAVAL SHIP RESEARCH AND DEVELOPMENT CENTER

Bethesda, Md. 20084

2

THE SHARP DATA BASE MANAGEMENT SYSTEM
USER INFORMATION MANUAL

by
Ben Wallis
Roy Hyde
Teri Blake

APPROVED FOR PUBLIC RELEASE: DISTRIBUTION UNLIMITED

COMPUTATION AND MATHEMATICS DEPARTMENT

AUGUST 1976
1st Revision

TM-188-76-1

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

DH

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS
BEFORE COMPLETING FORM

1. REPORT NUMBER
6 TM-188-76-1

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)
The SHARP Data Base Management System - User Information Manual,

5. TYPE OF REPORT & PERIOD COVERED

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)
10 Ben Wallis
Roy Hyde
Teri Blake

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS
David W. Taylor Naval Ship R&D Center
Bethesda, Maryland 20084

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS

12. REPORT DATE
11 August 1976

13. ~~NUMBER OF PAGES~~
205 12 212 p.

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)
14 DTNSRDC-TM-188-76-1

15. SECURITY CLASS. ~~(of this report)~~
Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE: Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

(Ships Analysis and Retrieval Program

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
SHARP
Data Base Management
Information Storage and Retrieval
COBOL Information System
Computer Program

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)
SHARP is a self-contained generalized Data Base Management System developed and operational on DTNSRDC's CDC 6700 SCOPE 3.4 Computer System. Data bases may be constructed, maintained, and queried interactively from remote terminals. English-like user oriented languages are used for data base definition, report definition, and query specifications. SHARP is also operational on the IBM 370, UNIVAC 70/45, UNIVAC 1108, and NOVA ECLIPSE Computers.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

JOB

387 682

THE SHARP DATA BASE MANAGEMENT SYSTEM

USER INFORMATION MANUAL

by

Ben Wallis
Roy Hyde
Teri Blake

AUGUST 1976
1st Revision

TM-188-76-1

TABLE OF CONTENTS

Page

Page

Page

1. INTRODUCTION

SHARP, Ships Analysis and Retrieval Program, is a generalized Data Base Management System (DMS) developed at the Naval Ship Research and Development Center (NSRDC), Carderock, Maryland. SHARP is operational on the CDC 6700 Computer System at NSRDC and the UNIVAC Series 70/45 Computer Systems at the Data Processing Service Center Pacific (DPSCPAC), San Diego, California and Data Processing Service Center Atlantic (DPSCLANT), Norfolk, Virginia. Current plans are to convert the system to the Burroughs 4700 Computer System at DPSCPAC.

SHARP allows on line and off line access to data bases. Users have general capability for defining, maintaining and interrogating data bases. User oriented English-like languages are used for both retrieval and report generation and are highly suited for interactive use from remote terminals.

Data Base Management Systems are normally classified as Self-Contained or Host Language Interface type. Host Language Interface Systems require some functions to be performed through application programs which interface with the DMS. These functions are often performed via subroutine-like calls from a user program to a DMS module. Updating is a typical function performed from a user program in a Host Language System. Also, Host Language Systems often do not have a "built in" query language for interrogating the data base. Self-Contained DMS's generally allow data bases to be defined, loaded, updated and interrogated without requiring application program interfaces. A built in query language and some data validation capability are essential for self-contained systems. Self-contained systems should, however, allow host language interfaces so that data processing beyond the capability of the DMS can be performed. This includes custom tailored reports and sophisticated data analysis.

SHARP is a self-contained DMS. It is designed to allow non-technical persons to define, build, maintain, and interrogate data bases without requiring application program interfaces. Application program interfaces are available if needed for user programs to access data in the data base and for generating special reports on data retrieved during a SHARP interrogation of the data base.

There is currently a variety of data base applications under the SHARP DMS. Some of these applications are listed below.

a) Technical Document (Bibliographic)
b) Ship Overhaul Experience
c) Ship Operating Experience
d) Computer Center Expense Accounting
e) Library Circulation
f) System Software Accounting
g) Ocean Science Research Project Information

SHARP is developed in the COBOL procedure language, which facilitates modifications and conversion to other computers.

1.1 Features

The major features of SHARP include:

- Generalized Data Base Definition
- Generalized User Oriented Format Check Capability
- Partially Inverted File Structure
- Variable Length and Variable Data Record Structure
- Ability to Change Data Base Definition without Rebuilding Data Base
- On Line File Maintenance with Free Form Direct Entry Capability
- English-Like user oriented Languages For Query Commands, Report Definitions, and Format Checking
- *Interactive Report Definition with* Ability to Catalog Defined Reports For Future Selection
- On line Interactive and Batch Retrieval
- On line Ad Hoc Computation Capability
- Host Language Interface
- Multi-level Sort Capability
- Hierarchical Tree Data Structure Capability
- Coded Data Element Translation Capability

1.2 System Installations

1.2.1 Naval Ship Research and Development Center (NSRDC), Carderock, Md.

SHARP is operational on the CDC 6700 computer system at NSRDC under the SCOPE 3.4 operating system and INTERCOM. INTERCOM, operating in conjunction with SCOPE, provides time sharing access to the system. This provides multiple users with simultaneous access to SHARP. The COMRADE Monitor, a general purpose monitor developed at NSRDC, is used to control the execution of SHARP.

SHARP requires 24,500 decimal words of core storage. This does not include the monitor which does not occupy core at the same time as an executing SHARP module.

1.2.2 Data Processing Service Center Pacific (DPSCPAC)

SHARP is operational on the UNIVAC series 70/45 at DPSCPAC under the DOS operating system and COS communications software.

Approximately 90,000 decimal bytes of core storage is required to execute SHARP. This includes approximately 20,000 bytes for a monitor which resides in core during execution of the system.

1.2.3 Data Processing Service Center Atlantic (DPSCLANT)

SHARP is under the same system configuration at DPSCLANT as DPSCPAC (described above).

2. DATA BASE STRUCTURE

A SHARP user data base comprises one or more data files. Each file consists of a collection of logical records. Each logical record contains information which describes a different entity in the data file. An entity could be a ship, a plane, a book, a person, a contract, a point in space, a task, etc. Thus, a data file could contain a collection of records, each of which describe a ship (entity). Each record would contain a set of descriptors or attributes which describe the entity (ship). The set of descriptors could be: Hull ID, ship name, weight, displacement, date entered service, ship type, class, etc. Each record must have a unique descriptor which allows it to be distinguished from other records in the data file. This descriptor is called the Record Key. In the above example the Record Key could be Hull ID. A record key which is one of the natural descriptors in the record, such as the one above, is called a Natural Key.

File determination in SHARP often depends upon user application interfaces. In general, what the user considered a file prior to entering the data into a SHARP data base can remain a file in the SHARP data base. For example, a user could have two personnel files. One could contain the usual personnel information such as name, grade or rank, job description, salary, number dependents, etc. The other file could be a training file containing specific information about each type of training taken by the employee. Under SHARP, information from both files could be put into a single file or the files could be maintained separately. Since a user may have application programs which access the files separately, it may be desirable to maintain them separately under SHARP. The data base described above is illustrated in Figure 2-1.

Job File

Training File

Record Key

Social Security Number

Name

Grade

Job Description

Number Dependents

Salary

Age

Code

Job Title

Previous Job Title

Previous Company

Previous Salary

Year Terminated

*Prior Work History

Social Security Number

Name

**

Date of Training

Type of Training

Subject Matter

Duration of Training

Degree or Certificate

Figure 2-1 - Personnel Data Base

* A set of data will be entered for each previous job held by the employee.

** A set of data will be entered for each type of training taken by the employee.

2.1 The SHARP Logical Record

The initial task in building a SHARP Data Base is Data Base Definition. This consists of defining for each data file the record key and all the descriptors or data elements which can occur in any one logical record. The set of data elements define a maximum entry for a logical record. An actual record in the data base may be comprised of any combination of the data base elements defined in Data Base Definition. Data elements are entered only if they are relevant for a given logical record. Thus, in the Job File of the Personnel Data Base described above, if an employee had no prior job, the four data elements comprising prior work history would not be entered in the record describing that employee.

A SHARP logical record may be any size up to a maximum of 10,000 characters.

2.2 Data Elements (Attributes)

Data Elements defined for a SHARP logical record may be either fixed or variable in length. A data element designated as variable may be from one character up to a specified maximum size. A fixed length data element is one which will always be a specified size whenever it is entered in a logical record. A noncomputational element may have a maximum size of 2000 characters.

2.2.1 Data Element Formats

Data Elements, in SHARP, may be defined as any one of the following formats:

a) Alphanumeric

b) Numeric

c) Alphabetic

d) Real

e) Integer

f) Free

2.2.1.1 Alphanumeric

An Alphanumeric data element is one in which each character is either an alphabetic letter (A-Z), a space or a number (0-9). The following examples illustrate:

a) ABC21

b) 29

c) AB 21

2.2.1.2 Numeric

A numeric data element is one in which all the characters are numbers. A numeric field is a coded field which does not represent a computed value or a measurement and is not to be used in a computation by the system. For example, a numeric field would not be a value representing a count. Social security number in personnel files and Federal Stock number in logistics files are examples of numeric fields.

2.2.1.3 Alphabetic

An alphabetic data element is one in which each character must be one of the letters A through Z or a space. The following examples illustrate:

a) ABC

b) AXR B

c) John Brown

2.2.1.4 Real

A Real data element is a computational number which has a decimal point embedded in the number. Real numbers may be either positive or negative.

When Real data elements are defined for a SHARP data base, the number of character positions to the right of the decimal point is specified. When the data is entered into the data base, the decimal may be either entered or omitted. If omitted, the decimal will be assumed by the System. On output reports, the decimal point will be re-inserted.

Real data values may be entered into a SHARP data base in a natural format. Leading zeroes may be entered or omitted. Positive numbers may be entered either with or without a leading plus sign. If more characters are entered to the right of the decimal than have been specified, the system will round to the proper size. Real numbers may be a maximum of 12 characters. This includes the sign position but does not include the decimal point. The following are examples of how data values may be entered for a real data element defined as being a maximum of 6 characters with 2 positions to the right of the decimal.

+14.62	
014.62	
0014.62	(System will assume 14.62)
001462	(System will assume 14.62)
1462	(System will assume 14.62)
-14.62	
-014.62	
14.617	(System will round to 14.62)
0.61285	(System will round to 0.61)
6.1	(System will assume 6.10)
-01462	(System will assume -14.62)
0.	

2.2.1.5 Integer

An integer data element is a computational number which represents a whole integer value. Integer numbers may be either positive or negative and may be a maximum of 12 characters including sign.

Integer values may be entered with or without leading zeroes. Positive numbers may optionally have a leading plus sign. The following examples illustrate:

121

0121

+121

+0121

-0121

-121

0

00

Both Integer and Real numbers represent computational values or measurements. Computational values are those which may require computational operations such as summing, averaging, computing percentage, etc. For example, such data elements as time measurement (hours), costs (dollars), and positional coordinates (X, Y, Z) would all be defined as either Real or Integer numbers.

2.2.1.6 Free Format

Data elements defined as having free format may consist of any valid characters in the character set of the computer system on which the data base resides.

2.3 Record Key

Each logical record in a SHARP data base file must have a unique Record Key which allows it to be located in the file. If one of the data elements which comprise the logical record will always have a unique data value for each record in the file, it may be assigned as the Record Key. This type of key is called a Natural Key. A typical natural key would be social security number in a personnel file.

If a data file does not have a natural Record Key, a unique key can be created by combining data elements in the record, using a data element in conjunction with a sequence number, or using an algorithm to generate a unique key. The user has the option of allowing the system to automatically generate a key by using an initial key number entered by the user as a base and adding one to the previous record value for each subsequent logical record. For example, a user could enter 1000 as the Record Key for the first record in the file. If no keys were entered for the next 50 records, the system would automatically assign the keys 1001, 1002, etc, up to 1050 for those 50 records. Then, if the user entered 2500 for the next record, followed by a series of records without keys entered, the system would enter 2501, 2502, etc, for that series of records. This allows a user to assign a series of numbers as logical Record Keys without having to enter numbers which are sequential.

In Data Base Definition, the data element format for the Record Key and the other data elements in each file is defined. The Record Key may be defined as any of the allowable system data element formats described in Section 2.2.1. Embedded blanks are allowed in Record Keys defined as Alphanumeric, Alphabetic or Free formats. Record Keys may be either variable or fixed in length and may contain up to a maximum of 30 characters.

2.4 Key Hashing

To conserve space in the SHARP data base, the internal Record Key is limited to 9 characters. If a user defined Record Key has a maximum size greater than 9 characters, a randomizing function is used to reduce the key to a unique 9 character internal key. The hashed key is normally transparent to a user. If a user desires to reference a record in a SHARP data base, the user defined key is always used regardless of whether or not key hashing is required. Key hashing is performed when new records are added to a SHARP data base. As each record is added, the user supplied key is passed through a randomizing algorithm which reduces it to the desired size.

During the hashing process, it is possible for 2 or more unique input keys to reduce to the same hashed key. Since each internal Record Key in a SHARP data base file must be unique, the duplicate keys are modified prior to insertion into the SHARP data base via a "tie breaker" process which changes the duplicate keys to unique ones.

2.5 Repeating Elements

A SHARP logical record may contain Repeating data elements. A Repeating data element is one which may have multiple data values in a logical record. The personnel data base described in Section 2.1 contains repeating elements in both files. In the Job File, each of the four data elements which comprise prior work history is a repeating element. Descriptors will be entered for each prior job held by the employee. For example, assume an employee had had three prior jobs. The following data values might be entered in the record:

Previous Job Title	Company	Previous Salary	Year Terminated
File Clerk	TRW	7000	1961
Computer Operator	Boeing	12000	1964
Computer Programmer	Computer Sciences	16000	1969

In the above example, each of the four data elements repeats three times. A repeating element may contain no entries or multiple entries in a logical record. Thus, if an employee had held no previous jobs, there would be no entries in the record for any of the four repeating elements. In the training file, the five data elements which comprise training information are each repeating. There will be one entry for each element for each type of training taken by the employee.

Next, consider a technical document data base, where each logical record describes a different book or document in a technical library. The logical record could contain such data elements as title, author, subject matter, report number, publication date, security classification, etc. Author and subject matter could be repeating elements as shown in the example below of a typical record in the data base.

Author	Subject Matter
John Jones	Data Base Design
Herb Smith	File Structures
Ann Brown	Data Definition Language Data Base Access

In this example, the document described in the record had three authors (Jones, Smith, and Brown). The subject matter is a list of key words which describe the subject content of the document. In this case, four key words were entered which describe the content of the documents. For this type of data base, the number of individual authors a document has could vary as could the list of key words used to describe the content of the document.

Repeating elements may have from 0 to 1000 entries per logical record.

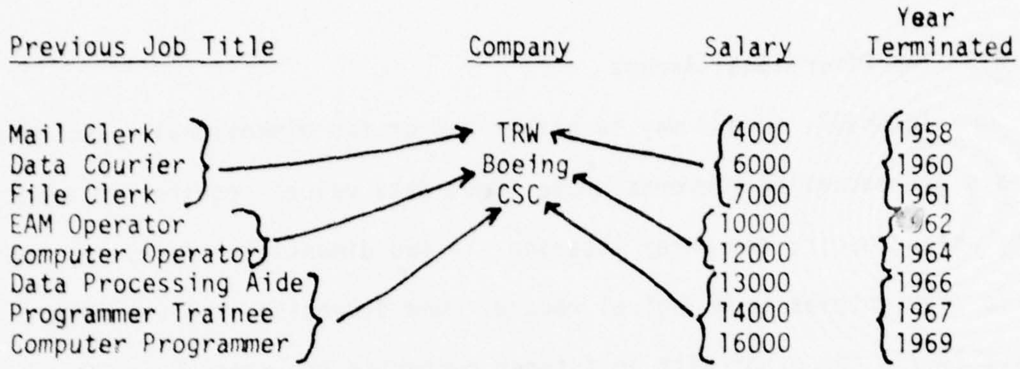
2.6 Arrays

A set of data values for a repeating element is called an array. Thus, there is an array for each data element defined as repeating. In the above technical document example, there is an author array and a subject matter array.

2.6.1 Parallel Arrays

In the example given in Section 2.5 on the job file, the four repeating elements defined are related. For each prior job held by the employee, there will be an entry in each of the four arrays. Thus, the four data elements, taken together, comprise an entity (prior job history). When information from multiple arrays are related in such a manner, they are said to be parallel. In parallel arrays, information in one array corresponds to information in one or more other arrays. If parallel arrays are such that they each have the same number of entries in a record and the nth entry in one array corresponds to the nth entry in the other arrays, they are said to be "even" parallel arrays. The parallel arrays described in the job file are even parallel arrays. Thus, the second entry in each of the arrays describes the second prior job in the record and the information (COMPUTER OPERATOR - BOEING - 12000 - 1964) combines to describe the prior job.

In the job file example given in Section 2.5, only one job was listed with each previous company where the employee worked. However, since an individual may have held more than one job at any of the previous companies, it may be desirable to list all the jobs held at previous companies. In such cases, it is not necessary to repeat the company name in the company array for each entry in the job title array. The information may appear as follows:



In this case, multiple entries in the job title array correspond to single entries in the company array. The arrays are parallel since they do contain corresponding information which combines to describe previous job history. The arrays are not even, however, since the n^{th} entry of the job title array does not correspond to the n^{th} entry of the company array. In this case, the two arrays are "uneven" parallel arrays. Of course, the three arrays job title, salary, and year terminated are even parallel arrays with respect to each other but uneven with respect to the company array. Note, that if the company name were repeated for each entry in the other three arrays, the arrays would then be even. However, this is unnecessary and wasteful of data base storage space.

2.6.2 Non-Parallel Arrays

In the technical document example given in Section 2.5, two arrays are defined; Author and Subject Matter. The two arrays do not contain related information. Thus, the author array could contain more or less entries without affecting the subject matter array. Such arrays which contain unrelated information are called non-parallel arrays.

2.6.3 Two Dimensional Arrays

In SHARP, arrays may be either one or two dimensional. Two dimensional arrays are repeating elements whose input data values require two associator tags which specify the array location. A two dimensional array may contain up to 2600 entries in a logical record. One dimension is given with a letter value (A-Z), the other with an integer number in the range 0 to 99. Thus,one dimension may have a maximum of 26 entries,the other 100. The user may decide which dimension is to be the row or column entry in the array.

When two dimensional array information is entered into a SHARP data base, the associator tags are specified. They may be entered in any convenient order without regard to tag sort sequence.

Any combination of array locations may be used for entering data values for a data element in a given logical record. No storage is reserved or used by the system for array locations for which data is not specified.

After entering array type data into a SHARP data base, users may reference specific array locations during both updating and queries by specifying the appropriate associator tags. Thus, data may be both retrieved and updated by array location.

Figure 2-2 illustrates a two dimensional array into which data values have been stored.

	0	1	2	3	4
A	145				
B		16		15	
C					182
D	120		121		

ARRAY DISTRIBUTION

Record Input For Repeating Element

INPUT DATA VALUE	INPUT ASSOCIATOR TAGS
145	A,0
121	D,2
16	B,1
15	B,3
120	D,0
182	C,4

Figure 2-2 - Two Dimensional Array

2.6.4 Associator Tags With Non-Repeating Elements

Associator tags may be used with non-repeating data elements. The tags may be used in whatever manner desired by the user. Either or both tags may be used with a given input data value.

The use of associator tags can minimize the number of data elements required in a user Data Base Definition. For example, assume a data base contains accounting information and among the information required are two associated values - amount and month of the year. The amount could be defined as a data element and the numeric associator tag could be used to express the associated month information. The following example illustrates:

Amount - 1520

Month - February

Data Base Input

Amount	Tag
1520	2

Associator tags may be referenced during both queries and updates. Thus, a user could search for all records in the data bank which had amounts greater than 1500 during February. Records containing the above information would satisfy the search.

2.7 Multiple File Data Bases

Users may define either single or multi-file data bases. In general, multiple files with the same Record Key can be consolidated into a single file data base. The Personnel data base described in Section 2 is such a data base since both files have the same key (social security no.). A consolidated data base will require less storage space since redundant information across files is eliminated. A consolidated data base may, in some cases, require a larger amount of computer processing time than processing the same data in a single file. Several factors such as user application interfaces, types of queries to be made, data redundancy, and data relationships should be examined prior to making a decision on file determination.

2.7.1 Application Interfaces

A user may have multiple files which are accessed by different application programs prior to entering the data into a SHARP data base. If the files are large and are processed serially by the application programs, consideration should be given to maintaining the files separately under SHARP, since some extra overhead will be required in an application program reading a large consolidated file rather than the individual file containing only the data of interest.

2.7.2 Data Redundancy

If a user has multiple files with the same-Record Keys which have multiple data elements in common, consideration should be given to consolidating at least some of the files in order to eliminate redundant data.

2.7.3 Query Processing Required

The type of query processing required may indicate whether or not files should be consolidated or maintained separately. If the data is such that a user would normally be interested in a certain segment of the data base, that segment should probably be defined as a separate data base file. For example, assume a user is considering whether to consolidate two files which contain information about ships. One contains Atlantic ships, the other Pacific ships. Also, assume that both files contain identical information about the ships. If there were users who in making queries, would be interested only in either Atlantic or Pacific ships but not both, the files should be defined as separate data base files under SHARP. The reason for this is that during a query, a user would always be specifying whether he is interested in the Atlantic or Pacific section of the file. Since there would be a large number of records contained in either, the internal processing required to separate the two segments consumes processing time not used if the two segments were in separate files.

2.8 Data Validation

Users may specify in Data Base Definition validity check information for each data element defined. As the data values are input during updates, the system will format check the data values in accordance with the user specified validity information. The types of validity check information which can be specified are given below:

a) A data element value must be a specified no. of input characters (Exact size check).

b) A data element must not exceed a specified no. of input characters (Maximum size check).

c) A data element must be a specified format (Alphabetic, Numeric, Alphanumeric, Real, or Integer).

d) A data element value must not contain spaces (No blanks check).

e) A data element value must be one of a specified list of values.

f) A data element value must be contained in a specified range of values or must be contained in one of a series of specified value ranges.

2.8.1 Subfield Validation

Users may specify validity check information for an entire data element value or a specified part of a data element value. For example, a user may specify that the first two characters of a data element be alphabetic and the next two characters be numeric. This is called subfield data validation. All the validity checks specified above may be used in subfield validation. In addition, users may specify that specific subfields be spaces or zeroes.

2.8.2 Boolean Relationships Among Validity Specifications

Validity specifications may be multiple for a data element and may have both "AND"and "OR" relationships. For example, a data element can be specified to be 6 characters long and be numeric. Also a specific data element or subfield could be specified as being alphabetic or all zeroes.

2.8.3 Textual Edit Validation

A special validation can be specified for textual type data elements so that extraneous information can be edited out of the data. This validation will remove any spaces between words in addition to the normal spacing. One space will be allowed between words in a sentence. Two spaces will be allowed between the end of one sentence and the beginning of the next.

2.9 Data Element Inversion

SHARP is classified as a partially inverted system. Any of the data elements defined in Data Base Definition may be designated for inversion. Inversion is a process which allows the System to use a direct access technique in determining the records in a file which satisfy a user query on the file. Data Elements which will be queried on by a user should be inverted.

Extra file storage is required for those data elements which are inverted. The amount of file storage can be extensive for large data bases. Therefore, data elements which are not to be queried on should not be inverted.

2.10 Hierarchical (Tree) Structure

Data elements may be defined in data base definition as having a hierarchical structure. This indicates that values for a particular data element represent a tree hierarchy.

Consider a personnel data base. The user may wish to define the data element "Job Skill" as a tree element. For each record (person) in the data base, the user would enter the most definitive job skill category as taken from a table of hierarchical job skills.

Fig. 2-3 illustrates a portion of a tree table constructed by the user for the job skill data element. Two main categories, computers and construction, are illustrated. Each main category is sub-divided into

```
COMPUTERS

          COMPUTER SOFTWARE

                    APPLICATIONS PROGRAMMING

                              COBOL PROGRAMMING
                              FORTRAN PROGRAMMING
                              ALGOL PROGRAMMING
                              JOVIAL PROGRAMMING

                    SYSTEMS PROGRAMMING

                              OPERATING SYSTEMS INTERNALS

                                        SCOPE INTERNALS
                                        OS-IBM INTERNALS
                                        DOS INTERNALS
                                        GECOS INTERNALS

                              COMPILER PROGRAMMING

                                        FORTRAN COMPILERS
                                        COBOL COMPILERS

                              TELECOMMUNICATIONS

                              DBMS INTERNALS

          COMPUTER HARDWARE

                    COMPUTER HARDWARE DESIGN
                    COMPUTER HARDWARE MAINTENANCE

CONSTRUCTION

          ROAD CONSTRUCTION
          BRIDGE CONSTRUCTION
          HOME CONSTRUCTION
                    CONDOMINIUM CONSTRUCTION
                    HOUSE CONSTRUCTION
```

Fig. 2-3 - A Typical Tree Structure-Data Element "Job Skill" In A Personnel Data Base.

as many lower categories as is required to contain the individual job skills entered for the employees.

Assume that a user, interrogating the data base, was interested in all persons having a job skill of Systems Programming. All persons (records) from the data base would be selected which had job skills of Systems Programming or any of the job skills which are subsets of it. For example, a person with a job skill of Scope Internals would be selected as would one with a skill of Compiler Programming or Telecommunications. Thus, a user can perform a broad (generic) search by querying on a category higher in the tree or a more narrow search by selecting a category lower in the tree. If a user queried on the skill "Cobol Compilers", only those persons with that specific skill would be selected, since there is no subset of Cobol Compilers in the tree.

An employee may have multiple skills entered in his personnel record which are unrelated in the tree hierarchy. Thus, an employee who had worked on computer hardware as a designer and who had been employed in construction as a bridge builder might have two job skills entered in his personnel record - Computer Hardware Design and Bridge Construction.

The tree capability for hierarchical data elements is advantageous because it minimizes the number of data element values which must be entered for a tree element in a record, thus reducing data base storage.

2.11 Coded Data Elements

Data elements in data base definition may be specified as being coded. Coded data elements are those where the user assigns codes to specific data element values for the purpose of reducing data base storage. Data elements which have large data values and have a small number of unique values compared to the data base size are good candidates for coded definition. For example, consider a 50,000 record data base containing the data element "Insurance Company", where there are 100 different companies. Each would be repeated many times throughout the 50,000 record data base (500 times each, assuming an even distribution and one insurance company per record). If the average data value size were 38 characters, the amount of storage for the raw data would be 1,900,000 characters (50,000 X 38). The coded definition capability allows the user to assign small unique codes for each unique data value (each insurance company). In this case, a 3 digit code is sufficient for the 100 different values. The amount of storage required would be 150,000 characters for the raw data (3 char. code X 50,000 records) plus the storage required for a coded translation table for the 100 different insurance companies. The coded translation storage would be 4100 characters (41 chars. for code plus entry X 100 entries). Thus, the total raw data required for the coded element in the data base would be 154,100 characters (150,000 raw storage plus 4100 for translation table). The storage savings would be 1,745,900 (1,900,000 minus 154,100) or a 92% reduction in storage.

For coded data elements, the system will automatically produce the data value rather than its assigned code on output reports. (e.g., For the above example, "Hartford Insurance Company" would be output rather than its assigned code.

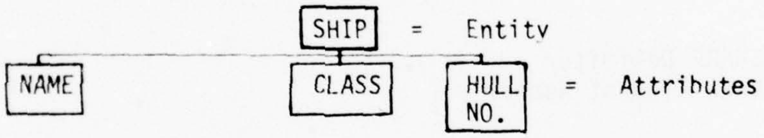
3. DATA BASE DEFINITION

SHARP is designed to process any type of data. Therefore, the user must identify what data elements comprise a data base and their characteristics before SHARP can accomplish the actual processing of the data. This process is called defining the data base.

3.1 Defining The Logical Record

SHARP utilizes the concept of logical records. The contents of one logical record can be described as an entity and all the information (attributes) that describes that entity.

Example:



The contents of one logical record can and often will differ from the contents of another logical record. This difference occurs when a certain attribute is present for one entity but not present for the other entity (s).

3.2 Data Element Definition

SHARP requires that certain information be provided for each data element that may appear in a logical record. This information is used to control the processing of data throughout the system.

3.2.1 Data Element Number

SHARP requires a unique data element number be assigned to each data element that is to be entered into the data base. The data element number can be any value within the range of zero through nine hundred ninety-nine with the following exceptions. Data element number zero can only be assigned to the record key (see Section 3.2.3). Data element number one is used by SHARP for recording the natural record key when the record is hashed due to the length of the data element identified as the record key (see Section 3.2.4). Data element number two is used by the system to identify the date the logical record was added to the file or when the content of the logical record was modified.

Example:

```
0)
|└──────── SHARP Delimiter
└───────── Data Element Number
```

3.2.2 Data Element Name

SHARP requires each data element be assigned a unique name. The acronym assigned can be any combination of characters A through Z, zero through nine and special characters - slash (/) and dash (-).

The acronym is limited to a maximum of ten characters in length. Embedded blanks are not allowed.

Example:

```
5)  TOTAL
6)  TRAN/CODE
7)  A1
```

Acronyms ending in the suffix "DATE" are assumed to be a numeric field containing a date (either YYMMDD or Julian YYDDD).

3.2.3 Record Key Assignment

SHARP requires that each record in a data base file be identified by a record key. The data value contained in the record key must be unique for each logical record. The record key may be an attribute whose data values are naturally unique for each file record (e.g. HULL ID for a ship file). If no such condition exists, the record key can be manufactured by combining multiple attributes to form a unique key or through the use of an algorithm such as combining an attribute with a sequence number or using a stand alone sequence number. Whenever possible, it is desirable to use one or more of the natural data elements to comprise the record key. The record key must be defined as data element number zero. When the chosen record key is greater than nine characters in length, the key will be hashed into a random set of characters. The natural value of the record key will be carried in the logical record as data element number one.

3.2.4 Record Key Hashing

SHARP utilizes a one to nine character record key field in the physical record. If the record key is greater than nine characters in length, SHARP will generate a random key by hashing the actual record key value. The hashed value is then used as the value for the record key. When a duplicate random number occurs, SHARP attaches a tie breaker to the random number, creating a unique random number.

Hashing is transparent to the user and automatic; therefore, the user is not required to specify hashing as part of the data definition.

3.2.5 Repeating Groups

SHARP provides the capability for processing a data element (s) that has multiple occurrences within a single logical record. A typical example is a "total" field occurring twelve times giving one total per month for a period of a year. Repeating data values are limited to a maximum of one thousand per data element per logical record. All occurrences for a single data element are processed under the control of a single data element definition statement.

Repeating groups are identified by placing the key word 'REPEAT' in the data definition statement.

Example:

```
0) TOTAL;REPEAT
```

Record keys can not be a repeating group.

3.2.6 Array Definition

SHARP provides the capability for specifying the relationship of occurrences between repeating groups. When a data element, identified as a repeating group, is also identified as an array, the user can control the updating and retrieval of the individual occurrences within each group.

3.2.6.1 Parallel Arrays

Parallel arrays are classified as two basic types.

Even Parallel Arrays occur when multiple data elements within a logical record have the same number of occurrences and the n^{th} entry of one repeating group is related to the same n^{th} entry of another repeating group.

Example:

Two data elements, each contains four entries -

Car Make	FORD	CHEV	OLDS	DODGE
Year of Make	1970	1974	1975	1972

Figure 3-1

In this example there is only one car of each make. Therefore, the year '1970' only describes the car make 'FORD'. Note that a single entry in the car array corresponds to a single entry in the year array.

Uneven Parallel Arrays occur when the number of occurrences in one repeating group is different from the number of occurrences in the other repeating group and corresponding entries are not always related.

Example:

Car Make	FORD			CHEV		
Year of Make	1970	1974	1975	1969	1971	1973

Figure 3-2

The array shown in Figure 3-2 is an uneven parallel array. While the number of occurrences in each data element is different, a relationship still exists. Occurrences one, two, and three of 'YEAR OF MAKE' are related to occurrence one of 'CAR MAKE'. Occurrences four, five and six of 'YEAR OF MAKE' are related to the second occurrence of 'CAR MAKE'.

3.2.6.2 Non-Parallel Arrays

Non-parallel arrays occur when the occurrences of one repeating group are not related to any occurrence of another repeating group.

3.2.6.3 One-Dimensional Arrays (see Figure 3-1, Section 3.2.6.1) are identified by specifying the keyword 'LINK' in the data definition statement for the repeating group. 'LINK' indicates that a tag will be assigned specifying the array location.

Example:

```
4)CAR/MAKE;REPEAT;LINK
5)YEAR/MAKE;REPEAT;LINK
```

3.2.6.4 Two-Dimensional Arrays (see Figure 3-2, Section 3.2.6.1) are identified by placing the keywords 'LINK'and'ROLE' in the data definition statement for the repeating group. 'LINK' and 'ROLE' are the tags which will specify the two dimensional array locations.

Example:

```
4)CAR/MAKE;REPEAT;LINK
5)YEAR/MAKE;REPEAT;LINK;ROLE
```

Data element four is a one-dimensional array, while data element five is a two-dimensional array. Note that element five is two-dimensional because of the way it relates to element four. In order to relate multiple values from element five to a single value in element four, two tags are required.

3.2.7 Multiple File Data Base

SHARP can not simultaneously access multiple data files. However, SHARP does have the capability for defining and processing multiple logical files within a single physical data base.

When the multiple logical file concept is utilized, a single data element must be used as the record key for all logical files. Section 3.4.2 contains an example of multiple data files within a single physical data base.

3.2.8 Validity Checking

SHARP is designed to provide certain data validation checks on incoming file maintenance transactions prior to updating the data base. Data validation is optional; therefore, SHARP will accept any value and length (maximum of 2000 characters for non computational elements) for a data element if data validation specifications are not included as part of the data definition.

3.2.8.1 Size Specification

The size option indicates to Sharp the exact length of a data element. SHARP insures that the length of the data value in the file maintenance transaction matches the length specified in the size option. When a difference is detected, SHARP rejects the transaction as invalid.

Example:

```
4)TOTAL;SIZE=6
```

3.2.8.2 Maximum Size Specification

The maximum size option can be used in place of the size option. When the maximum size option is specified , SHARP will accept any data value that does not exceed the length in the maximum size statement.

Example:

```
4)TOTAL;MAXSIZE=6
```

The size or MAXSIZE specification is required for all data elements and it is conventional to enter it immediately following the element acronym name as shown above.

3.2.8.3 Class Specification

SHARP is designed to test a data value for the correct class. There are six classes of data in the SHARP system.

Free: The free option is default. When no other class is specified, any combination of characters will be accepted as valid data value.

Alphabetic: When the alphabetic class is specified, a valid data value can be any combination of characters A through Z, or spaces. Alphabetic is denoted by the keyword ALPHAB.

Example:

```
4)NAME;ALPHAB
```

Numeric: When the numeric class is specified, a data value can be any combination of characters zero through nine. A numeric field is a coded non-computational field. Numeric is denoted by the keyword NUMERIC.

Example:

```
4)SOC/SEC/NO;SIZE=9;NUMERIC
```

Alphanumeric: When the alphanumeric class is specified, the data value can be any combination of characters A through Z, zero through nine, or spaces. Alphanumeric is denoted by the keyword ALPHAN.

Example:

```
4)CODE-A;ALPHAN
```

Real: This option, when specified, identifies the data element as a computational number which has an embedded or assumed point. The location of the decimal point must be identified. Real numbers can be a maximum of twelve characters (excluding decimal point but including the sign). The keywords REAL and DEC define real elements.

Example:

```
4)TOTAL; REAL; DEC=2
                  └----Decimal point location.
                       Counted from the unit position,
                       indicates two places to right of decimal.
```

Integer: When the integer option is specified, the data element is assumed to be a computational whole number (no decimal point). Integer elements are defined by the keyword INTEGER.

Example:

```
4)TOTAL; MAXSIZE=5; INTEGER
```

3.2.8.4 Range Specification

SHARP has the capability for testing a data element for a range of values. Any value within the low and high limits of the range is considered a valid data value. The keywords RANGE or RANGES are used to denote range specifications.

Example:

```
7)MONTH; RANGE=1/12
```

The low and high limits of the range can contain any combination of characters. Each limit is restricted to a maximum of ten characters. The range specification is inclusive and includes the end points. Thus, a value of 12 would satisfy the above range specification. A maximum of five ranges can be specified for a single data element. Multiple ranges indicate an 'OR' condition. The value must fall into at least one of the ranges to satisfy the validity check.

Example:

```
5)CARD-CODE; RANGES = 17/21, 27/31, OR 40/49
6)TYPE; RANGES = A/D OR J/R OR V/Z
```

3.2.8.5 Value Specification

SHARP is designed with the capability for testing a data element for specific values. One or more values can be stated for a single data element. Multiple values in a value statement are treated as an 'OR' condition.

Example:

```
4)TRAN/CODE; VALUES = A, C, OR D
5)DELET; VALUE = D
```

The maximum number of allowable values is ten. Each value can contain a maximum of ten characters.

3.2.8.6 NOBLANKS Specification

SHARP provides the capability for specifying that a data element not contain embedded spaces.

Example:

7)MONTH; ALPHAB; NOBLANKS

Note that ALPHAB alone does not exclude blanks.

3.2.8.7 Subfield Specification

SHARP provides the capability for dividing a data element into subfields for the purpose of data validation. A maximum of five subfields are allowed for a single data element.

Subfield validation is normally performed on data elements with a class of free, alphabetic, numeric or alphanumeric.

The class, range and value options can be specified for one or more subfields with these exceptions. The value option is limited to five values, each limited to a maximum of five characters. The range option is limited to one range per subfield with the size of the limits restricted to five characters each.

Subfield specifications are defined by placing the keyword CHAR(S) and the subfield location in the data definition statement.

Example:

```
10)DATE; CHARS n/m     where,

                       n=beginning left-hand character
                         number of subfield

                       m=ending right-hand character
                         number of subfield
```

10)DATE; CHARS 1/2

Validation specifications for the subfield must immediately follow the key word and field location. If there are validity specifications for both the whole data field and one or more subfields, the following order should be used for input-- validity specifications for whole element, 1st subfield specification, validity specifications for 1st subfield, 2nd subfield specification, validity specifications for 2nd subfield, etc.

Example:

```
5)DATE; SIZE 6; CHARS 1/2; RANGE 01/12; CHARS 3/4; RANGE 01/31; CHARS 5/6;
  VALUE = 75
```

```
6)CODE-A; MAXSIZE = 14; NOBLANKS; CHARS 1/2; VALUES = AD, AC, OR AM.
```

The above data definition statement identifies "DATE" as a six digit field. The first and second characters must be in the range of one through twelve inclusive, characters three and four must be in the range of one through thirty-one, and characters five and six must be the value seventy-five.

3.2.8.8 Inversion Specification

SHARP utilizes the partially inverted file concept. Therefore, the user is allowed to determine what data elements will be inverted.

Inversion must be performed upon data elements that will be the subject of query searches.

Inversion is specified by placing the key word "INVERT" in the data definition statement.

Example:

```
6)TOTAL; INVERT
```

3.2.8.9 Validity Specification For Spaces and Zeroes

Validity testing for spaces and zeroes may be done through the use of special keywords - SPACE, SPACES, ZERO, ZEROES. These keywords are used in the same manner as keywords which indicate class (NUMERIC, ALPHAB, ETC.)

Example:

```
4) CODEA; SIZE = 4; CHAR 1;
   NUMERIC; CHAR 2; SPACE

5) CODEB; SIZE = 4; CHAR 1; ALPHAB;
   CHARS 2/3; ZEROES

6) CODEC; MAXSIZE = 4; CHAR 1;
   VALUES = A,B, OR C; CHARS 2/3;
   SPACES; CHAR 4; ZERO
```

The testing for spaces and zeroes is normally relevant only at the subfield level.

3.2.8.10 Textual Edit Specification

Users may specify that a textual type data element be edited when input into a SHARP data base. The editing will remove extraneous spaces between words and insert proper spacing around commas and periods. The textual edit is indicated by inserting the keyword TEXT in the data element definition.

Example:

```
4)ABSTRACT; MAXSIZE = 1500; TEXT
```

3.2.8.11 The Boolean "OR" in Validity Specifications

Unless otherwise specified, compound validity specifications have an implied "AND" relationship. For example - ALPHAB; NOBLANKS - indicates that the field must contain alphabetic characters and not contain embedded blanks. An "OR" relationship may be specified by inserting the conjunction "OR" between keyword specifications.

Example:

```
4)  CODEA; SIZE = 4; CHAR 1; NUMERIC
    OR VALUE = R

5)  CODEB; SIZE = 1; ALPHAB OR ZERO

6)  CODEC; SIZE = 1; ALPHAB OR RANGES
    = 1/3, 5/6, OR 8/9

7)  CODED; MAXSIZE = 9; CHARS 6/7; NUMERIC
    OR SPACES; CHAR 8; VALUE = -
```

Note that the OR may be used at both the whole and subfield level. In element 4, character one must be either numeric or have a value of "R". The OR may be used with special keywords (SPACES and ZEROES). Note that the "OR" separating the keywords ALPHAB and RANGES has a different function than the "OR" separating the range values. The boolean "OR" should always be entered between keywords (e.g. NOT RANGES = 1/3, 5/6, OR 3/9 OR ALPHAB). A maximum of five "OR" levels may be used at the whole element level. Only one level is allowed at the subfield.

3.3 Coding Conventions

SHARP allows free form formating of the data definition statement with a few exceptions.

The data element number must be the first parameter entered and must be followed by a right parenthesis.

The data element name must immediately follow the data element number.

Keywords such as; RANGE, VALUE, LINK, ROLE, REPEAT, NUMERIC, ALPHAB AND ALPHAN may appear in any order and must be separated by a semi-colon.

Example:

```
5)CODE; INVERT; NUMERIC
     is the same as
5)CODE; NUMERIC; INVERT
```

Spaces between words or delimitors may be omitted or of a variable number.

Example:

```
5)TOTAL;    SIZE = 6;      INVERT
            is equal to
5)TOTAL;SIZE=6;INVERT
            or
5)TOTAL;    SIZE                             (card 1)
            =                                (card 2)
            6;   INVERT                      (card 3)
```

3.3.1 File Identification

SHARP requires that the statement preceding each file definition for a user data base be a file identification statement.

Example:

```
ID filename
|     L--------filename can be any combination of non blank characters
|              with a maximum length of six characters.
L--------------constant
```

3.3.2 Standard Data Formats

3.3.2.1 Dates

SHARP has standard data formats for dates. Dates should be defined as one of the following formats -

FORM	EXAMPLE	INTERPRETATION
YYMMDD	750629	29 JUNE 1975
YYMM	7407	JULY 1974
YY	73	1973
YYDDD	74033	2 FEB 1974
YDDD	3021	21 JAN 1973

Dates are defined with year being left most in the field, since for sort and retrieval purposes year is the most important part of the field.

3.3.2.2 Personal Names

Personal names are entered with last name being first or left-most in the field.

Example:

Jones John Q
Wilson A. B.

3.3.3 Card Format

SHARP utilizes the first seventy-two columns of each card for data definition specifications. Card columns seventy-seven through eighty are used for process control and sorting.

The first seventy-two card columns are used for recording data specifications (see Section 3.3). Columns seventy-seven through seventy-nine contains the data element number. Column eighty is a card sequence letter which in conjunction with the data element number provides a unique key for sorting.

When the data specifications exceed the limits of a single card, the additional specifications are continued on the following cards. When the need for continuation cards arises, card column eighty is required to contain an ascending character value for sorting the multiple cards into the correct processing sequence.

Example:

```
c.c.1                                  c.c.77
5)DATA;SIZE=2Ø4;                          ØØ5A
  INVERT;                                 005B
  ALPHAN                                  005C
```

3.4 Examples of Data Base Definitions

3.4.1 The following example illustrates a personnel file. The logical record contains twelve data elements. The record key will be employee number.

Employee Number is a unique number assigned to each individual. The data value can range from one through five hundred and six. The data will be entered as a variable length value (no leading zeroes).

Employee Name is a thirty character field. Data values will be variable length (no trailing spaces) and will be tested for an alphabetic content.

Date Employeed is a six character field in the format of YYMMDD. Each subfield within date employed will be validated for content such as - the month subfield will be tested for a numeric value within the range of one through twelve, the day subfield will be tested for a numeric value within the range of one through thirty one, the year subfield will be tested for a range of forty through seventy-five.

Social Security Number is a nine character field containing the individual social security number. All data values must be numeric and contain exactly nine digits.

Employee Salary is a four character field. Data values represent dollars (thousands). Each data value will be a real computational number and variable in length (no leading zeroes). Each number will represent the salary divided by 1000 (e.g. 15.26 = 15,260.)

Department Code is a two character field. Valid department codes are ten, twenty and thirty. Department Code is a repeating group showing current and past department assignment. This field will be a one-dimensional array.

Job Titles is a thirty character field containing position descriptions such as clerk, accountant, computer programmer, etc. Job title will be a two-dimensional array parallel to the Department Code array. Multiple entries in the Job Title array may correspond to single entries in the Dept. Code array, indicating that an employee held more than one position in a given department. The two arrays are uneven. Data values for Job Titles will be variable length and alphabetic in content.

Employee Age is a two character field containing the individual's age. Data Values must be a two digit positive integer number within the range of sixteen through sixty-five.

Sick Leave is a three character integer field containing the number of sick leave hours earned. Data values will be of variable length and contain both positive and negative integer numbers.

Job Description is a ninety character field. Data values will be narrative in content and variable in length. Job Description is a two-dimensional array parallel to the Job Titles and Department Code arrays. It is even with respect to the Job Titles array and uneven with respect to the Department Code array.

All data elements, except for name, social security number and sick leave will be the subject of query searches and will require inversion.

PERSONNEL FILE DATA DEFINITION

```
ID PERS
0)EMP/NO; MAXSIZE 3; INVERT; NUMERIC; RANGE = 1/506                   0A
2)PRO/DATE; SIZE = 6; INVERT; NUMERIC                                 2A
3)NAME; MAXSIZE 30; ALPHAB                                            3A
4)EMP/DATE; SIZE 6; CHARS 1/2; RANGE = 40/75;                         4A
  CHARS 3/4; RANGE = 1/12; CHARS 5/6; RANGE = 01/31; INVERT           4B
5)SALARY; MAXSIZE 4; REAL; DEC 2; INVERT                              5A
6)DEPT/CODE; SIZE = 2; VALUE = 10, 20, OR 30; REPEAT;                 6A
  LINK; INVERT                                                        6B
7)J/TITLE; MAXSIZE 30; ALPHAN; REPEAT; LINK; ROLE;                    7A
  INVERT
8)SSN; SIZE = 9; NUMERIC                                              8A
9)AGE; SIZE = 2; INTEGER; INVERT; RANGE = 16/65                       9A
10)SPEC/SKIL; MAXSIZE = 20; NOBLANKS; INVERT                         10A
11)SICK/LV; MAXSIZE 3; INTEGER                                       11A
12)JOB/DESC; MAXSIZE = 90; TEXT; REPEAT;LINK;                        12A
  ROLE                                                               12B
```

3.4.2 Technical Document File

Figure 3-3 illustrates the data base definition for a Logistics Technical Document Data Base. The data elements include bibliographic information such as title, author, publication date, security, etc.

```
ID NAVLIS

0)   ACCESSION; MAXSIZE = 5; INVERT; CHAR 1; NUMERIC OR VALUE = R;
     CHAR 2/4; NUMERIC; CHAR 5; NUMERIC OR SPACE

1)   TITLE; MAXSIZE = 192

2)   DOC/NO; MAXSIZE = 23; INVERT; REPEAT

3)   DESCRPTN; MAXSIZE = 1500

4)   TYPE; MAXSIZE = 15; ALPHAB; INVERT

5)   PUBL/DATE; MAXSIZE = 6; NUMERIC; INVERT; CHARS 1/2; RANGE = 00/75;
     CHARS 3/4; RANGE = 01/12; CHARS 5/6; RANGE = 01/31

6)   SECURITY; SIZE = 1; INVERT; VALUES = U, C OR S

7)   NUM/COPY; MAXSIZE = 2; NUMERIC

8)   PAGES; MAXSIZE = 8; ALPHAN

9)   LIB/LOC; MAXSIZE = 9

10)  AD/NO; SIZE = 8; ALPHAN; INVERT

11)  VOL/NO; MAXSIZE = 2; ALPHAN

12)  PUB/UIC; MAXSIZE = 5; NUMERIC

13)  AUTHOR; MAXSIZE = 23; INVERT; REPEAT

14)  ORG/NAME; MAXSIZE = 120

15)  SOURCE; SIZE = 6; NUMERIC; INVERT

16)  REV/DATE; MAXSIZE = 6; NUMERIC; CHARS 1/2; RANGE = 00/75

17)  ADO/TASK; MAXSIZE = 10; ALPHAN; REPEAT; INVERT

18)  PSS/REF; MAXSIZE = 5; NUMERIC; REPEAT; INVERT

19)  KEYWORD; MAXSIZE = 36; REPEAT; INVERT

20)  INDEX; MAXSIZE = 50; REPEAT; INVERT

21)  CMD/SEC; MAXSIZE = 12; ALPHAB
```

Figure 3-3 - Data Base Definition Logistics Technical
Document Data Base

3.4.3 Multiple File Data Base Definition

Users may define both single and multiple file data bases. Consider a user with two separate related personnel files which he wishes to put under SHARP. The first file contains three data elements - employee number, name, and pay number. The second file contains employee number, training schools attended, class scores and date of school. The record key for both files is employee number. Both files may be consolidated into a single file data base or both files may be maintained separately. Section 2.7 gives the various factors which should be considered before deciding whether to consolidate data files.

To define a consolidated single file data base, a user would simply define a single file which contains a composite of the elements from both files as shown below.

```
ID  PERSNL
0)EMP/NO; MAXSIZE 4; INVERT; CHARS 1/3;
  NUMERIC; CHAR 4; VALUE = P OR T
1)NAME; MAXSIZE 30; ALPHAB
2)PAY/NØ; SIZE 6; INVERT
3)SCHOOL; MAXSIZE 20; ALPHAB; REPEAT; LINK;
  INVERT
4)SCORE; SIZE 2; RANGE = 00/99; REPEAT; LINK
5)DATE; SIZE = 6; REPEAT; LINK
```

The three elements SCHOOL, SCORE and DATE are parallel arrays containing related information.

To define the two files separately under SHARP, the user would enter two separate file definitions as shown below.

```
ID  PERSNL1
0)EMP/NO;   (Validity check info.)
1)NAME;           "      "     "
2)PAY/NO/;        "      "     "

ID  PERSNL2       "      "     "
0)EMP/NØ;         "      "     "
1)SCHOOL;         "      "     "
2)SCORE;          "      "     "
3)DATE;           "      "     "
```

Note that the separate files may have the same data element number assigned for two different elements (SCHOOL and NAME). In the composite file each element must have a unique number assigned.

3.4.4 Defining a Tree Data Element

If a data element has a hierarchical tree structure (see Sec. 2.10), it is defined by including the keyword TREE in the data element definition.

Example:

```
15)SKILL; MAXSIZE = 36; INVERT; REPEAT; TREE
```

3.4.5 Defining a Coded Data Element

If a data element is coded (see Sec. 2.11), it is defined by including the keyword CODED in the data element definition.

Example:

```
16)INS-COMP; MAXSIZE = 3; INVERT; CODED
```

4. FILE MAINTENANCE

SHARP provides the capability to create and maintain a data base. The data definition, described in Section 3, identifies that data elements are to be processed and the validation specifications for editing the file maintenance transactions.

4.1 Creating a Data Base

A SHARP data base can be created simply by coding an add transaction for each logical record to be entered into the data base.

4.2 Changing a Data Base

A SHARP data base can be changed by adding additional records, changing or adding a data element within an existing logical record or by deleting a data element from a logical record or deleting an entire logical record.

4.3 Coding of File Maintenance Transactions

4.3.1 File Identification

SHARP requires the first statement of a set of transactions to be the file identification statement.

Example:

```
ID file name
|     L---------- as identified in the data definition
L--------------- constant
```

4.3.2 Transaction Types

SHARP has three types of transactions: an add transaction which is used to insert a new logical record into the data base; a change transaction which is used to add a data element into an existing logical record or to replace an existing data value with another value; and a delete transaction which is used to remove a data value from an existing logical record or to delete an entire logical record from the data base.

4.3.2.1 Adding a Logical Record

An add transaction is denoted by a literal "A" as the first character of the transaction.

Example:

```
          A
```

The record key value must immediately follow the "A".

Example:

```
          A CVA61
          |  L-------- record key
          L------------- constant
```

Data elements that make up the logical record being added then follow the record key, each element preceded by its data element number as defined in data base definition.

Example:

```
          A CVA61 3) ABC 4) 125, etc.
                         || L---------- data value
                         |L------------ delimiter
                         L------------- data element number
```

4.3.2.2 Changing a Logical Record

A logical record can be modified by entering a change transaction. A change transaction is denoted by a literal "C" as the first character of the transaction.

Example:

```
C
```

The record key must immediately follow the literal "C".

```
C CVA61
|   L---- record key
L-------- constant
```

The data element(s) to be updated or added are entered following the record key.

Example:

```
C CVA61 10) DEF 11) 4567
                |  |  L---- data value
                |  L-------- delimiter
                L----------- data element number
```

4.3.2.3 Deleting a Logical Record or a Data Element from a Logical Record

A delete transaction is denoted by a literal "D" as the first character of the transaction.

Example:

```
D
```

The record key must immediately follow the literal "D".

Example:

```
D CVA61
|    L----- record key
L---------- constant
```

When data elements are being deleted rather than an entire logical record, the data elements are entered following the record key.

Example:

```
D CVA61 7) 4520
|   |     L------ delimiter
|   L----------- record key
L--------------- constant
```

For this type of delete transaction, it is not necessary to enter the element value to be deleted. Any non-blank character will suffice.

4.3.3 Coding Conventions

SHARP allows the user to use a free form format for entering data with the following exceptions.

- The transaction code must be the first character in the transaction.
- A card or line of input is limited to seventy-two characters.
- The end of a logical transaction is denoted by the literal "$END".
- The end of a set of transactions is denoted by the literal "$STOP".
- Multiple data elements may be placed in a single line or card with one or more spaces between the data elements.
- When a data value is too large to fit on one line or the remaining portion of a line, the value can be continued into the next line. The last character of the value must be in position seventy-two with the next character of the value being placed in position one of the next card.

4.4 Entering of Data Values

The presence of a data element in a transaction is denoted by the data element number (see Section 3.2.1), a right parenthesis and the data value (not required for deletes).

Example:

```
D CVA61 3) ABC
|   |   || L------ data value
|   |   |L--------- delimiter
|   |   L---------- data element
|   L-------------- record key
L------------------ transaction code (Delete)
```

4.4.1 Methods of Entering Data Values

SHARP allows data elements to be described as variable length (maxsize option). When entering the data value for such an element, only the significant characters are required.

Example: Data element four is described as maxsize equals thirty.

```
4) JOHN DOE
      or
4) JOHN WILLIAM SMITH
```

When the size option is used, the data value entered must be of the length specified in the size option.

Example: Data element five is described as size equals six.

```
5) 000001
```

Example: Data element five is described as maxsize equals six; therefore, leading zeroes are not required.

5) 1

When entering data elements described as "real", the decimal point may be entered or omitted.

Example: Data element four is described as REAL; MAXSIZE = 4; DEC 2

4) 10

is equal to

4) .10

4) 1234

is equal to

4) 12.34

When entering data elements described as INTEGER or REAL, the positive sign is optional.

Example: Data element seven is described as INTEGER, maxsize equals six.

7) +199

is equal to

7) 199

Negative values must be preceded by the negative sign (minus).

Example:

7) -199

8) -17.42

When entering data values for elements described as repeating groups (REPEAT), multiple occurrences may be entered separated by semi-colons.

Example:

10) al; b2; 3c

In the above example, the value "al" will be placed in the first occurrence of data element ten. The value "b2" will be placed in the second occurrence and the value "3c" will be placed in the third occurrence.

When a specific occurrence of a repeating group is to be added or changed, the relative occurrence is denoted by preceding the value with n-1 semi-colons.

Example:

10) ;;;A4

In the above example, the value "A4" will be placed into the fourth occurrence of data element ten.

When entering data values for elements identified as one-dimensional arrays (LINK), the associator tag or occurrence factor must be stated as part of the entry. The associator tag is entered immediately following the delimiter "L". A comma separates the data value from the tag specification.

Example:

```
11) 179, L = A
    |   \/   L----- associator tag
    |    \---------- delimiters
    L-------------- data value
```

The value 179 will be placed into array location A. LINK associator tags must be one of the characters "A" through "Z". Multiple values may be entered in the same method used for REPEAT only data elements.

Example:

11) 179, L = A; 180, L = B; 200, L = K

In the above example, the data value 179 will be placed into location A, value 180 will be placed into location B and value 200 will be placed into location K.

When entering values for data elements described as two-dimensional arrays (LINK, ROLE), two associator tags must be entered.

Example:

```
12) 175, L = A, R = 1
                |       ↑--- second associator tag
                ↑----------- first associator tag
```

13) 195, L = A, R = 1; 196, L = A, R = 2; 200, L = B, R = 4

The preceding transaction, when processed, will produce a two-dimensional array with these values.

ROLE

LINK	1	2	3	4
A	195	196		
B				200

An important difference exists between elements defined as REPEAT and those defined as REPEAT; LINK. In the former case, there is no need to enter tag information with the repeating data values. The data values are not associated or related to information in other arrays. In the latter case, tag information is required, usually to relate the array values to values in other arrays (parallelism).

4.5 Sample Transactions

The following transactions will produce a data base with three logical records using the data base definition for the personnel file described in Section 3.4.

```
A   56   3) SMITH JOHN Q    4) 740126    5) 10.34
         6) 20, L = A;  30, L = B
         7) OPERATOR, L = A, R = 1; PROGRAMMER, L = A, R = 2;
            BRANCH HEAD, L = B, R = 1     $END

A    7   3) SMITH MARTHA E  4) 700429  5) 900
         6) 10, L = A; 20, L = B
         7) CLERK TYPIST, L = A, R = 1;
            SECRETARY, L = B, R = 1;
            STENOGRAPHER, L = B, R = 2
         8) 455568748    $END

A  207   3) JONES MARY R  4) 752904  5) 7.89
         6) 10, L = A    7) DATA CLERK, L = A, R = 1
         $STOP
```

In the records above, elements 6 and 7 are parallel arrays. Element 6 is Department Code and element 7 is Job Title. The associator tags relate the information between arrays. In record 56, John Smith was an Operator and a Programmer in Department 20 and a Branch Head in Department 30.

In record 7, Martha Smith was a Clerk Typist in Department 10, and a Secretary and Stenographer in Department 20. In record 207, Mary Jones is a Data Clerk in Department 10.

The following examples illustrate various change and delete transactions on the above three records.

Example: In record 56, various data elements defined in data base definition were not entered when the record was created. Change the record by adding some of these data elements to it.

```
C  56   8) 488289683  9) 49   11) 145   $END
```

Example: In record 7, change the 2nd entry in the Department Code array from 20 to 30.

```
C   7   6) 30, L = B   $END
```

Example: In record 7, delete the entry for social security number.

```
D   7   8) X    $END
```

Example: Delete record 207 from the data base entirely.

```
D   207   $END
```

Example: Change the record key in record 207 from 207 to 208.

```
D   207   $END
A   208   3) JONES MARY R  4) 752904
    5) 7.89  6) 10, L = A  7) DATA CLERK, L = A, R = 1, $STOP
```

Note in the last example that to change a record key, the normal change procedure cannot be used. The record must be deleted, then added back with the correct key.

In an update transaction, it is not necessary that data elements be entered in sequence by element number.

```
Example:      A  110   1)AB  4)220  3)17  $END
              C  107   3)22  1)AM   $END
```

4.6 The Double Dollar Sign Delimiter

The user may avoid splitting transaction data values between lines by using a double dollar sign delimiter as shown below.

```
A  1421   1)23   2)750421   3)SURVEY OF $$
EXISTING LOGISTICS SYSTEMS  4) JONES A. B. $END
```

The double dollar sign signals the end of data for that line. Data pickup begins in character position one of the next line. The technique will eliminate extra spaces inadvertently being entered between words in textual type data elements. This is particularly helpful in terminal updating where the end of line character position is not shown. When updating by punched cards, where the character position is shown, the technique is not necessary.

4.7 Converting Existing Files to SHARP

Users may have existing files not under SHARP which they wish to convert to the SHARP DMS. In this case, it is not necessary to build the SHARP files by re-entering all the data as add transactions. A conversion module exists which will automatically convert existing files to the standard SHARP internal transaction format which is then used to create the SHARP data base files.

4.8 Validity Checking Input Transactions

When data is entered in the standard SHARP transaction format, it is validity checked in accordance with the validity specifications entered in the Data Base Definition. Data which fails the format checking will be flagged as invalid. Only data elements found in error will be rejected. If the user is entering the transactions on line, he will be allowed to re-enter any data values rejected. See Section 8.2 for an example of on line file maintenance.

4.9 Entering Multiple Transactions for Same Record

Multiple transactions may be entered simultaneously for the same record. The following examples illustrate:

```
A  121    1)AC  2)MV  3)240  $END
A   50    1)AR  2)MP  3)279  $END
C  121    2)MC  $END
D   50  $END
C  145    3)250  $END
D  145    2)MN  $STOP
```

The transactions are performed in the order in which they are entered. Thus, record 121 is added, then later element 2 is changed in record 121 from MV to MC. Record 50 is added, then it is deleted. Element 3 is changed in record 145, then element 2 is deleted from the same record.

4.10 Constructing a Tree Element Table

If a data element has a tree structure (see Sec. 2.10), a file consisting of a table of values in the tree hierarchy must be constructed prior to entering values from the table into a SHARP data base.

Following the data base definition, where the data element is defined as a tree element (see Sec. 3.4.4), the user constructs a table of tree values versus hierarchical codes. The example in Sec. 2.10 (Fig. 2-3) illustrates job skill from a personnel data base as a typical tree element.

The user first determines the maximum number of levels of hierarchy required to contain the required values. The maximum number of levels allowed for the assigned hierarchical code is 12. Also, the assigned hierarchical code must not exceed 12 characters. Assume, in the case of the job skill tree in Figure 2-3, that 6 hierarchical levels are required. A table of hierarchical codes might be assigned as shown below.

```
AAA  COMPUTERS
        BA  COMPUTER SOFTWARE
              CA  APPLICATIONS PROGRAMMING
                    DA  COBOL PROGRAMMING
                    DB  FORTRAN PROGRAMMING
                    DC  ALGOL PROGRAMMING
              CB  SYSTEMS PROGRAMMING
                    DA  OPERATING SYSTEMS INTERNALS
                          EA  SCOPE INTERNALS
                          EB  OS-IBM INTERNALS
                          EC  DOS-INTERNALS
                          ED  GECOS INTERNALS
                    DB  COMPILER PROGRAMMING
                          EA  FORTRAN COMPILERS
                          EB  COBOL COMPILERS
                    DC  TELECOMMUNICATIONS
                    DD  DBMS INTERNALS
        BB  COMPUTER HARDWARE
              CA  COMPUTER HARDWARE DESIGN
              CB  COMPUTER HARDWARE MAINTENANCE

AAB  CONSTRUCTION
        BA  ROAD CONSTRUCTION
        BB  BRIDGE CONSTRUCTION
        BC  HOME CONSTRUCTION
              CA  CONDOMINIUM CONSTRUCTION
              CB  HOUSE CONSTRUCTION
```

The above example illustrates 5 levels of hierarchy for the job skill "computers" and 3 levels for "construction." The internal hierarchical code for each entry consists of all the codes which define the entry's tree relationship.

Example:

```
AAA - COMPUTERS
AAA  BA - COMPUTER SOFTWARE
AAA  BA  CB - SYSTEMS PROGRAMMING
AAA  BA  CB  DC - TELECOMMUNICATIONS
AAA  BB  CB - COMPUTER HARDWARE MAINT.
AAB  BC  CB - HOUSE CONSTRUCTION
```

Note, in the example, that 3 characters are used for the assigned code for the first level (computers) and 2 characters are used for each lower level. For 6 levels and 12 characters, the "characters per level" breakdown would be 3,2,2,2,2,1. The user establishes the number of code characters for a level by determining the maximum number of table entries required for the level. Any combination of the 26 letters and 10 numbers may be assigned as codes. Thus, at the main level, 46,656 entries (36 times 36 times 36) may be assigned. "Computers" and "Construction" are two such entries at the main level. At the second level, 2 characters are allowed. Thus, 1296 (36 times 36) skills may be entered under each main level entry.

4.10.1 Loading a Tree Element Table

After the tree table has been determined and the codes assigned, the user may load the tree file. The loading and updating of the tree file is performed by entering the appropriate add, change, or delete transactions preceded by command statements which identify the tree element and its structure. The following example illustrates the technique used to load table entries for the tree element "skill" shown in the example in Sec. 4.10.

```
NEW CONTEXT DATABASE PERSNL DFID SKILL;
STRUCT  3 2 2 2 2 1;
A  AAA  COMPUTERS;
A  AAABA  COMPUTER SOFTWARE;
A  AAABACA  APPLICATIONS PROGRAMMING;
A  AAABACADA  COBOL PROGRAMMING;
A  AAABACADB  FORTRAN PROGRAMMING;
A  AAABACADC  ALGOL PROGRAMMING;
   .       .       .
   .       .       .
   .       .       .

A  AABBCCA  CONDOMINIUM CONSTRUCTION;
A  AABBCCB  HOUSE CONSTRUCTION;
EOJ;
```

In the above sequence, the first statement is a CONTEXT statement. This statement identifies the applicable data base (PERSNL) and the data element (SKILL) which has a tree structure. The word "NEW" is required only for the initial loading of the file. On subsequent update runs, it is omitted. Other options are available with the CONTEXT statement and will be given later.

The second statement is the STRUCT statement which defines the number of characters used for each level of the hierarchy. This information is required so that listings of the tree table can be output in the indented manner shown in Sec. 4.10. The STRUCT statement is required only with the initial loading of the tree file.

Following the STRUCT statement are the transactions for loading the tree table. Each transaction is an add transaction and is denoted by the letter "A" followed by the assigned code and the table entry value.

An end-of-job statement (EOJ) terminates the job. Note, that all statements in the sequence are terminated by semi-colons.

Table entries may be changed or deleted as illustrated by the following example.

```
CONTEXT DATABASE PERSNL DFID SKILL;
C    AABBA  HIGHWAY CONSTRUCTION;
D    AABBB;
EOJ;
```

The first transaction above is a change transaction (Code C). The table entry whose code is AABBA is changed to "Highway Construction". The second transaction is a delete (Code D). The table entry whose code is AABBB is deleted. Note, that the table entry value is not required with a delete transaction.

4.10.2 Tree Table Synonyms

Synonyms for tree table entries may be specified after an initial entry is entered in a tree table.

Example:

```
CONTEXT DATABASE PERSNL DFID SKILL;
A    RCC   DATA BASE MANAGEMENT SYSTEM;
ASYN  RCC  DBMS;
ASYN  RCC  DBM;
```

In the above example, the code "RCC" has been assigned to the table value "DATA BASE MANAGEMENT SYSTEM". The synonyms "DBMS" and "DBM" are assigned the same code. Thus, the three values are synonymous. Users could query on any of the three values and get the same results.

Synonyms previously entered in a tree table may also be deleted in a similar manner.

Example:

```
          DSYN  DBMS;
```

Note, that in deleting synonyms the assigned code is not entered with the transaction.

4.10.3 Non-Hierarchical Tree Table Values

It is possible to enter table values which are non-hierarchical (no sub-levels) into a tree table containing hierarchical entries.

Example:

```
NONHIER CONTEXT DATABASE PERSNL DFID SKILL ENTRYTYPE TWELVENON;
A   PERSONNEL TRAINING;
A   ZOO KEEPING;
A   BEE KEEPING;
EOJ;
```

In the above example, the CONTEXT statement has been expanded to specify that non-hierarchical table entries will follow (NONHIER). Also, "ENTRYTYPE TWELVENON" indicates that explicit codes will <u>not</u> be entered and that the first twelve non-blank characters from the table value will be used for the assigned code. Thus, the code which will be assigned for PERSONNEL TRAINING will be "PERSONNELTRA". This technique of implicit code assignment is convenient for non-hierarchical values since the assigned code does not have to be hierarchical. Explicit codes may be assigned by omitting the ENTRYTYPE from the CONTEXT statement and using the format shown in Sec. 4.10.1. An alternate implicit entry type available is

"ENTRYTYPE TWELVE" which indicates that the first twelve characters of the table value will be used for the assigned code (including blanks).

4.10.4 Listing a Tree Table

Some useful listings for tree tables are available via a "LIST" command. The LIST command, which may be run alone or with an update run, allows tree tables to be printed sequenced by either code or table value.

Examples:

```
LIST ALL TREE IN PERSNL;
```

This list will produce for each tree element in data base PERSNL a list of assigned code versus table value. The list will be sequenced by assigned code. Hierarchical table entries will be listed in the indented format shown in Sec. 4.10. This list does not output synonyms, only initial entry table values.

```
LIST SKILL IN PERSNL;
```

This list is the same as above, except only for the data element skill.

```
LIST DECODED TREE IN PERSNL;
```

This list will produce for each tree element all the tree table values, including synonyms, and their assigned code. The listing will be sequenced by table value.

```
LIST ONLYHIER SKILL IN PERSNL;
```

This list produces only the hierarchical entries sequenced by code in an indented format.

```
LIST NOHIER SKILL IN PERSNL;
```

This list produces only non-hierarchical entries sequenced by assigned code.

```
LIST DECODED ONLYSYN SKILL IN PERSNL;
```

This list will produce only the synonym entries sequenced by table value.

```
LIST ONLYHIER SKILL RANGE AA TO RR IN PERSNL;
```

This list produces hierarchical entries sequenced by code in the specified range.

4.10.5 Updating a Data Base Containing Tree Elements

After the tree table file has been constructed, a user may enter tree values into a data base. The tree values are entered in the normal SHARP update format.

Example:

```
ID PERSNL
A  455-56-8736  1)JONES J.Q.  2)TECHNICIAN
 15)FORTRAN PROGRAMMING; BRIDGE CONSTRUCTION
$END
```

In the above example, assume that data element 15 is "skill", a tree element in the PERSNL data base. In this case, two data values are entered into the record. Both values have been previously entered into the tree table file constructed for the data base. Sec. 3.4.4 illustrates the data element definition for the tree element. Note, that the validity check information applies to the table value and not the assigned code. A maximum of 50 characters are allowed for tree table values.

4.11 Constructing a Coded Translation Table

If a data element is defined as coded (Sec. 2.11), a file consisting of a table of assigned codes versus table values must be constructed. The codes are assigned in any convenient manner and are treated as non-hierarchical by SHARP. The number of characters used for the assigned code normally depends on the expected size of the table. A table of 500 entries requires only a two-character code, since 1296 assignments can be made with a two-character code (36 times 36).

4.11.1 Loading a Coded Translation Table

The loading and updating of a coded translation table is performed in the same manner as with a tree table. The appropriate add, change or delete transactions are entered preceded by a CONTEXT statement which identifies the appropriate data base and data element.

Example:

```
NEW CONTEXT DATABASE CLAIMS DFID INCOMP MAX 60;
A  01  GOVERNMENT EMPLOYEES INSURANCE COMPANY;
A  02  MUTUAL OF OMAHA;
A  03  STATE FARM;
A  04  ALL STATE;
A  05  VOLKSWAGON OF AMERICA INSURANCE COMPANY;
EOJ;
```

The above example illustrates the loading of a coded translation table for the data element "Insurance Company" (INCOMP) in the data base "CLAIMS" Five entries are loaded into the file. Changes and deletes to the table are made in the same manner as with a tree file.

In the CONTEXT statement, MAX indicates the maximum size that a table value may be. MAX is required only for coded elements and only for the initial loading of the file. Table values for coded elements may be a maximum of 100 characters.

4.11.2 Listing a Coded Translation Table

A coded translation table can be printed with a LIST command, sequenced either by code or table value.

Examples:

```
LIST INCOMP IN CLAIMS;
(Outputs table sequenced by assigned code)

LIST DECODED INCOMP IN CLAIMS;
(Outputs table sequenced by table value)
```

4.11.3 Updating a Data Base Containing Coded Elements

After a coded translation file has been created for a data base containing coded data elements, the coded information may be entered into the user data base. In the updating, the assigned codes are entered, not the table values. Thus, the format check information specified in the data element definition applies to the assigned code (see Sec. 3.4.5 for an example of data element definition).

Examples:

```
ID CLAIMS
A   KEY1   1)760421 . .. 16)02 ... $END
A   KEY2   1)760427 .... 16)04 ....$END
```

In the preceding examples, data element 16 is the coded element "Insurance Company" (INCOMP). The codes for "Mutual of Omaha" and "All State" have been entered into two data base records. On output reports, the system will convert the codes to the appropriate table values (Insurance company names).

5. QUERIES

Users may interrogate SHARP data bases using an English like command language. In the language, questions may be phrased which specify the desired selection criteria. Those records in the data base which satisfy the selection criteria will be retrieved and output. With each question entered, users may specify the information to be output from each record via a Print statement or a Report statement which identifies a previously defined Report. The desired sort sequence of the selected records may also be specified with each question.

Multiple questions may be entered simultaneously in both the interactive and batch modes. A maximum of 75 questions may be entered at one time. With each question, a different Sort, Print, or Report specification may be entered.

Each question entered will be analyzed for errors. The query values entered will be checked to ensure that the format is the same as that defined in the Data Base Definition for the specified data elements. The questions will also be checked for Query language syntax errors. If errors are found, appropriate diagnostics will be output. During an interactive query, error diagnostics for each question are output immediately to the user. The user may then re-enter the information correctly prior to entering the next question.

Following the successful inputting of a user query, the system will determine for each question the number of data base records (hits) which satisfied the specified selection criteria. For on line queries, the number of hits for each question will be output for user inspection on the terminal. The user will then specify for each question whether or not the hit records will be printed on line on the user terminal, off line on a batch printer, or not reported at all. The number of hits for a question is often a factor in the decision to print a report on line or off line. If the number of hits is large, a user may decide to defer the printing of the report to an off line printer. If the number of hits is small or an off line printer is not readily available, the user may elect to print the report on line. The user may only be interested in the number of records which satisfy a question and not data from the hit records. In this case, the user will specify that the hit information not be output.

On line queries may be entered in both an interactive and batch terminal mode.[1] In the interactive mode, the user enters a query and monitors the job during the entire execution of a query job from query input to report output. In the batch terminal mode, the query is entered and then the user is asked to specify whether the query results are to be displayed on line at the terminal or to an off line printer. The user then logs off the terminal and the query run is entered into a batch queue where it is executed in batch mode alonq with other batch jobs. If the user specified off line output, he may retrieve his output results at an off line printer after the execution of the batch run. The amount of time from input of query to execution of the batch run varies according to computer backlog. The average has been 30 minutes to one hour at NSRDC. If the user specified on line output in the batch terminal mode, he returns to the terminal after the period of time required for batch execution of the job. If the job has been run, the system responds with the number of hit records for each question. From this point on, the system is

[1] Batch terminal mode currently available only on CDC 6700 version of SHARP at NSRDC.

used exactly as in the interactive mode. For each question, the output report may be printed on line, printed off line, or not printed at all. Figure 5-1 illustrates the steps required for a user for both interactive and batch on line queries.

Users may desire to submit query jobs off line via a job control punched card deck. In this case, the query questions may be punched on cards using the same command language used for on line queries. Off line and batch terminal on line queries are run whenever the user expects a large amount of printer output or the time the user can wait for the output is relatively long. Also, a data base may be too large to be maintained on line. In this case, it must be accessed either off line or batch terminal on line since only on line data bases stored on permanent disk files may be accessed interactively. Jobs run interactively have a faster throughput time than batch jobs.

USER PROCEDURE
INTERACTIVE TERMINAL

1. Log On
2. Select Data Base File
3. Enter Query
4. Receive number of hits for each question. Specify output disposition for each question.
5. If on line output specified, receive output at terminal.
6. Log off
7. If off line output was specified, receive output from off line printer.

USER PROCEDURE
BATCH TERMINAL

1. Log On
2. Select Data Base File
3. Enter Query
4. Specify on line or off line output.
5. Log off
6. (After Job Execution) If off line output was specified, receive output from off line printer.
7. Log back on only if on line output was specified.
8. Receive number of hits for each question. Specify output disposition for each question.
9. If on line output was specified, receive output at terminal.
10. Log off
11. If off line output was specified, receive output from printer.

Figure 5-1 - Interactive and Batch Terminal Query Procedures

5.1 Query Construction

5.1.1 Data Element Names (Acronyms)

In order to communicate to the system the data elements which the user desires to query on, the acronyms assigned in Data Base Definition are used. For example, in a personnel file, the acronyms assigned to social security number, job title, and Number of Dependents might be SS/NO, JOB-TITLE, and NO-DEPNDTS, respectively. These acronymns would be used in referencing those data elements in a query and are called search acronyms.

5.1.2 Search Criteria

In a query the search criteria qualifies the search. For example to state that a data element (JOB-TITLE) be equal to a data value, the search criterion would be "equal". Figure 5-2 gives a complete list of available search criteria. Each criterion has one or more combinations of allowable syntax combinations which can be used in specifying the criterion in a query. For example, to specify the criterion "equal", the user could enter any of the following in the query:

IS
EQ
EQUAL
EQUALS
=

CRITERIA	* ENTER CODE
**Is; Equal to	IS IS EQ TO IS EQUAL TO IS EQUALS TO IS = TO
Not Equal To	IS NOT IS NOT EQUAL TO IS NE TO
Less Than (but not equal to)	IS LT THAN IS LS THAN IS LESS THAN IS < THAN
Less Than or equal to	IS LE TO IS LESS OR EQUAL TO IS LESS THAN OR EQUAL TO
Greater Than (but not equal to)	IS GR THAN IS GT THAN IS GREATER THAN IS > THAN
Greater Than or Equal to	IS GE TO IS GREATER OR EQUAL TO IS GREATER THAN OR EQUAL TO
Between but not equal to	IS BN TO IS BETWEEN BUT NOT EQUAL TO IS BETWEEN
Between or Equal to	IS BE TO IS BETWEEN OR EQUAL TO
Between or Equal to higher value	IS BH IS BETWEEN OR EQUAL HIGHER
Between or Equal to lower value	IS BL IS BETWEEN OR EQUAL LOWER

** Default Search Criteria, used when no criteria is specified

* Underlined Codes are optional

Figure 5-2 - Table of Search Criteria

Note that underlined codes are optional and are allowed for readability and to allow most combinations of words that naturally occur when expressing a criterion in English grammar. For example, the "equal" criterion could also be expressed by any of the following:

IS EQ TO
EQ TO
IS =
EQUAL TO

5.1.3 Search Values

The search value is that value which the search acronym is compared to in the search. For example, if a personnel file is to be searched for all persons whose job title is programmer, the data element JOB-TITLE is compared on equality (EQ) to the search value PROGRAMMER. Search values are entered in the formats defined in Data Base Definition for the corresponding data elements.

5.1.4 Query Format

A query is composed of one or more questions. Each query is introduced by the Syntax - QUERY XX, where XX is a two or three character user identification tag such as personal name initials. Each question is introduced by the conjunction IF and is terminated by either a semi-colon or a question mark. Each question may contain one or more independent clauses (sentences) which contain as a minimum a search acronym, a search criterion, and one or more search values. A sentence is introduced by one of the conjunctions - IF, AND, OR, OR IF. Each sentence within a question except the last may be optionally terminated with either a semi-colon or question mark. The last sentence in a question must be terminated by either a semi-colon or question mark. The Syntax $END is entered following the last question in a query:

Examples:

```
QUERY BW
IF JOB-TITLE IS PROGRAMMER;

IF JOB-TITLE EQ PROGRAMMER OR OPERATOR?

IF NO-DEPNDTS IS GR THAN 3
OR SALARY LESS THAN 20000;

IF NO-DEPNDTS IS 2 OR 3;
AND JOB-TITLE IS PROGRAMMER OR OPERATOR;
AND CODE IS 1882?
$END
```

The above query consists of four questions. The first question contains one sentence which consists of an acronym (JOB-TITLE), a search criterion (IS), and one search value (PROGRAMMER). The second question contains one sentence, but there are two search values (PROGRAMMER and OPERATOR). This is called a compound search value. The third question contains two sentences each of which has one search value ("3" in sentence one, "20000" in sentence two). Note that the two sentences are joined by the conjunction OR. Question four contains three sentences with a total of five search values (two in sentence one, two in sentence two, and one in sentence three). Note the conjunction OR in question four is used differently than the OR in question three. In question three, OR introduces a sentence. In question four, it is used to join compound search values in both sentence one and sentence two.

The query language syntax closely approximates the way the questions would be asked in the English language by someone who wished to search a data base. In question 1, all persons (records from personnel file) would be selected whose job title is programmer. In question 2, persons whose job title is either programmer or operator would be selected. In question 3, persons with either more dependents than 3 or salaries under 20000 would be selected. In question 4, persons with either two or three dependents and who are either programmers or operators and who are in Code 1882 would be selected. Thus a programmer in Code 1882 with three dependents would be selected, while an operator in Code 1882 with four dependents would not be selected. Note that AND introducing a sentence imposes additonal restrictions on the search.

5.1.4.1 Format Variations

There are multiple ways in which compound search values may be expressed in the SHARP query language. The following examples illustrate:

```
1) IF SUBJECT IS RADAR
   OR SUBJECT IS SONAR
   OR SUBJECT IS TRACKING;

2) IF SUBJECT IS RADAR OR SONAR
      OR TRACKING;

3) IF SUBJECT IS RADAR, SONAR, OR
      TRACKING;

4) IF SUBJECT IS RADAR, SONAR OR
      TRACKING?

5) IF SUBJECT IS RADAR OR SONAR
   OR SUBJECT IS TRACKING;
```

All of the above examples are acceptable. The syntax used for questions 3 and 4 is recommended, however, since it more closely approximates the preferred English equivalent. The acronym SUBJECT must be entered exactly as defined in the Data Base Definition. Any of the optional search criteria equivalent to IS may be entered (see Figure 5-2, Section 5.1.2). The search values entered (RADAR, SONAR, TRACKING) will be matched by the system against the data values entered for SUBJECT in the file records. Thus, it is important that the search values be spelled exactly as they are entered in the data file.

5.1.4.2 Search Values With Embedded Spaces

Search values may have embedded blanks or may consist of multiple words. In either case, the spaces must be entered exactly as they would appear in the data values in the data base.

Examples:

```
IF SUBJECT IS SHIP HULLS;
IF HULL/ID IS SSBN 609;
```

In question 1 above, SHIP HULLS is a single search value but consists of two words separated by a space. In question 2, SSBN 609 is a single search value with an embedded space. Embedded blanks and periods are not allowed in search acronyms.

5.1.4.3 Format Restrictions

Queries are entered in a free form format. Multiple spaces are allowed between words, however, at least one space is required. Spacing on each side of punctuation marks is optional. Any number of words may be entered on a line. On punched card input, only the first 72 characters are used. Any number of lines may be used to enter a question. A maximum of 15 sentences and 25 search values may be entered per question. Search values may not exceed 50 characters in length (including embedded blanks). A maximum of 75 questions may be input at one time (interactive and batch mode).

Examples:

```
1)  Correct      QUERY JJS     IF   HULL/ID          IS
                    ASR 27    ;
2)  Incorrect     IFHULL/ID  IS  ASR 27;
3)  Correct       IF  HULL/ID  IS  ASR 27, DLG 6, OR  DDG 5;
4)  Incorrect     IF  HULL / ID   IS  ASR 27;
5)  Correct       IF   HULL/ID  IS  =        TO   ASR 27
                  ;
6)  Incorrect     IF  HULL/ID  IS  ASR 27    ORDDG5?
7)  Correct       IF  HULL/ID  IS  ASR 27   OR
                          DDG 5      ?
```

Question 2 above is incorrect because at least one space was not entered between IF and HULL/ID. Question 4 is incorrect because embedded blanks are not allowed in acronyms. Question 6 is incorrect because at least one space was not entered between the conjunction OR and the search value DDG 5. In general, it is not necessary to memorize a set of rules about spacing. If the spacing is entered as it would be if typed in a letter, the spacing will normally be correct.

5.1.4.4 Compound AND/OR Logic

A question may require both AND and OR used as conjunctions introducing sentences. In this case it is necessary to know which conjunction "dominates" in the logic grouping. In this case AND dominates OR.

Example:

```
IF JOB-TITLE IS PROGRAMMER
AND NO-DEPNDTS GR THAN 3
OR SALARY LESS THAN 20000;
```

In this case, if OR were dominant, records would be selected where SALARY is less than 20000 regardless of whether any other conditions were met. However, since AND is dominant, the logic would be grouped as follows:

```
IF JOB-TITLE IS PROGRAMMER
AND  [NO-DEPNDTS GR THAN 3
      OR SALARY LESS THAN 20000]
```

It can be seen that for a record to be selected, JOB-TITLE must be PROGRAMMER and in addition either of two conditions must be met -- NO-DEPNDTS must be greater than 3 or SALARY must be less than 20000. Thus, a programmer with 4 dependents making 21000 would be selected.

The above logic can be applied for grouping multiple levels of AND/OR logic.

Example:

```
IF NUM1 IS 10
OR NUM2 IS 20
AND NUM3 IS 15 OR 16
OR NUM4 IS 21
AND NUM5 IS 12;
```

The above would be grouped as follows:

```
IF      [NUM1 IS 10    ]
        [OR NUM2 IS 20 ]
AND     [NUM3 IS 15 OR 16]
        [OR NUM4 IS 21   ]
AND     [NUM5 IS 12];
```

For a record to be selected in the search, each bracket must be "satisfied". The first bracket can be satisfied in either of two ways -- if NUM1 IS 10 OR NUM2 IS 20. The second bracket can be satisfied in either of three ways -- IF NUM3 IS 15 OR 16 OR NUM4 IS 21. The final bracket can be satisfied in only one way -- NUM5 must be 12. The following table illustrates the selection process.

NUM1	NUM2	NUM3	NUM4	NUM5	RECORD SELECTED
10	30	40	21	12	YES
10	20	15	21	13	NO
10	21	17	21	12	YES
10	20	17	22	12	NO
20	20	16	25	12	YES

5.1.4.5 OR Dominance (OR IF)

In cases where both AND and OR are used as conjunctions introducing sentences, the OR condition may be made dominant by using OR IF.

Example:

```
IF JOB-TITLE IS PROGRAMMER
AND NO-DEPNDTS GR 3
OR IF SALARY LESS THAN 20000;
```

In this case, the logic grouping is as follows:

```
       IF   [JOB-TITLE IS PROGRAMMER]
            [AND NO-DEPNDTS GR 3    ]
    OR IF   [SALARY LESS THAN 20000 ]
```

The use of the OR IF changes the logic such that a record is selected if SALARY is less than 20000 regardless if other conditions are met. In this case either bracket may be satisfied and a record will be selected. Thus a PROGRAMMER with four dependents making 21000 would be selected.

Next, consider an example with AND, OR, and OR IF used.

Example:

```
IF NUM1 IS 10
AND NUM2 IS 15 OR 16
OR NUM3 IS 20
OR IF NUM4 IS 21;
AND NUM5 IS 12;
```

The grouping is as follows:

```
   IF [ NUM1 IS 10 ]
   AND  [NUM2 IS 15 OR 16]
        [OR NUM3 IS 20   ]
 OR IF  [NUM4 IS 21     ]
        [AND NUM5 IS 12 ]
```

In this case, the first AND dominates the first OR and the OR IF dominates both AND's. Thus, a record is selected if the last bracket is satisfied regardless of the conditions met by the first two brackets. The following table illustrates the selection process:

NUM1	NUM2	NUM3	NUM4	NUM5	RECORD SELECTED
11	17	25	21	12	YES
10	16	21	21	13	YES
10	17	25	21	13	NO
10	17	20	22	12	YES
11	15	20	22	13	NO

5.1.4.6 Queries on Repeating Elements

A repeating data element may have multiple data values entered in a data base record. Queries may be made on multiple data entries for a single data element. Consider a file which describes books. Each record describes a different book in the file and includes the data element personal author. Since a book may have multiple authors, the data element AUTHOR could be defined as repeating in Data Base Definition. If a user wished to locate a book written by multiple authors, the following query might be entered:

IF AUTHOR IS SMITH AND JONES;

In this case, all books would be located which were written by Smith and Jones. The search does not mean that the only authors must be Smith and Jones, only that Smith and Jones must be included among the authors. Thus, a book written by three authors would be selected if Smith and Jones were among the three authors. This type of search is meaningful only if the data element queried on is repeating.

"OR" type searches on repeating elements are satisfied if any of the data values entered satisfy the search.

Example:

IF AUTHOR IS SMITH OR JONES;

AD-A055 581   DAVID W TAYLOR NAVAL SHIP RESEARCH AND DEVELOPMENT CE--ETC   F/G 9/2
THE SHARP DATA BASE MANAGEMENT SYSTEM - USER INFORMATION MANUAL--ETC(U)
AUG 76   B WALLIS, R HYDE, T BLAKE

UNCLASSIFIED   DTNSRDC-TM-188-76-1   NL

2 OF 3
AD
A055 581

In this case a book would be located if any of its authors were either Smith or Jones.

5.1.4.7 Ranging Searches

Ranging search capability allows data value ranges to be specified rather than single data values.

Example:

```
IF TOTAL IS BE   25/160;
```

The ranging search criteria BE means "between or equal" (see Search Criteria table in Section 5.1.2). In this question, all records will be selected where TOTAL is in the range 25 to 160 inclusive. The two range values are separated by a slash. Records which had any of the following entries for TOTAL would be selected in the search -- 25, 26, 27, , , 160.

To specify a ranging criteria that is between but not equal to the specified values, the following could be entered:

```
IF TOTAL BETWEEN   25/160;
```

In this case, records with values for total in the range 26 to 159 inclusive would be selected.

To specify a ranging search which is between or equal to the lower range value, the following could be entered:

```
IF TOTAL BL   25/160;
```

In this case, records with values for total in the range 25 to 159 inclusive would be selected.

To specify a ranging search which is between or equal to the higher range value, the following could be entered:

```
IF TOTAL  BH   25/160;
```

In this case, records with values for total in the range 26 to 160 inclusive would be selected.

Compound search values may be specified in ranging searches as in other type searches.

Example:

```
IF TOTAL BETWEEN  17/28, 35/42,
            78/91,  OR  117/185;
```

In this example, records would be selected if values for total were in any of the four specified ranges.

In entering ranging values, spacing is optional on each side of the slash. Either no spacing or one or more spaces may be entered between the slash and the data values.

Ranging searches may be performed on alphanumeric or alphabetic type data elements as well as numeric, integer, or real data.

Examples:

```
IF CODE-A  IS  BE  AB/AR;
IF CODE-B  IS  BE  AR21/AR53;
```

In the first question, all records would be selected where CODE-A was any of the values -- AB, AC, AD, , , AQ, AR. In the second question, all records would be selected where CODE-B was in the specified range -- AR21, AR22, , , AR52, AR53.

In the alphanumeric collating sequence, alphabetics are lower in value than numerics. Thus, to specify a range which would include both alphabetics and numerics in the same range, the alphabetic should be specified as the first value.

Example:

```
IF   CODE   IS   BE   A/9;
```

In this case, all records would be selected where CODE was any of the values -- A, B, C,,, Z, 0, 1, 2,,, 8, 9.

5.1.4.8 Queries on Integer and Real Numbers

The same variations in format is allowed for queries on integer and real numbers as is allowed in entering the values into a SHARP data base. Any of the variations shown in Sections 2.2.1.4 and 2.2.1.5 is allowed. Search values may be entered either with or without leading zeroes. Positive values may be entered either with or without a leading plus sign. Real numbers may be entered either with or without the embedded decimal point. The following are examples of queries on an integer number (COUNT; MAXSIZE = 5) and a real number (OVH/HRS; MAXSIZE = 6; DEC 1).

```
1)    IF COUNT GR   25
      AND OVH/HRS  LT   105.1?

2)    IF COUNT GR    00025
      AND  OVH/HRS LT  00105.1;

3)    IF COUNT  GR    +025
      AND  OVH/HRS  LT   +105.1?
```

```
4)    IF COUNT  GR    025
      AND OVH/HRS  LT    1051;

5)    IF  COUNT  GR   25
      AND  OVH/HRS  LT    001051;

6)    IF COUNT  GR    0025
      AND   OVH/HRS  LT    105.09
```

When real numbers are entered without the embedded decimal, the system employs the Data Base Definition to determine the proper position of the decimal internally. It is recommended for queries, however, that the decimal be punched for better readability. Note that in question 6, the internal value will be rounded to 105.1 because of the corresponding Data Base Definition (1 character to the right of the decimal).

Variations are <u>not</u> allowed in entering search values for data elements other than integer and real. For example, consider the following data element definition --

```
7) CODE; MAXSIZE = 6; NUMERIC; INVERT
```

If the data value for CODE were entered into a record as 007695 and a user were searching for records with that code, it would be improper to enter the search value without the leading zeroes.

Example:

```
Correct      IF CODE IS  007695;
Incorrect    IF CODE IS   7695;
```

5.1.4.9 Prefix Search

Queries may be made on the beginning (left most) characters of a data element. This is often done when a data element format represents some kind of hierarchy. Such a search is called a prefix search. For example, assume that a company was divided into the following structure:

```
Company
   Departments      - 20
       Divisions    - 9
           Branches - 9 maximum
```

The company is divided into twenty departments. Each department is divided into nine divisions and each division consists of one to nine branches. Next assume that a hierarchical code was entered in the company's personnel file which indicated where each employee worked (which department, which division in that department, and which branch in that division). If an employee worked in department 18, division 7 in that department, and branch 4 within division 7, his code would be entered 1874. If a user, in a query, wished to locate all employees in department 18, the following query would be entered: (Assume acronym is CODE)

```
IF PREFIX CODE IS 18;
```

To locate all employees in department 18, division 7, the following would be entered:

```
IF PREFIX CODE IS 187;
```

To locate all employees in department 18, division 7, and branch 4, the following non-prefix search would be entered:

```
IF  CODE  IS  1874;
```

The following examples illustrate variations in the use of the prefix search.

```
1)    IF  PREFIX  CODE  IS  BE  11/19?

2)    IF  PREFIX  CODE  IS  11  OR  12
      AND NO-DEPNDTS  IS  LESS THAN 4;

3)    IF  PREFIX  CODE  IS  BE  07/10, 12/15,
          OR  17/19?

4)    IF  PREFIX  SUBJECT  IS  SHIP  HULL;

5)    IF  PREFIX  SUBJECT  IS  WEAPONS  AND
      SUBMARINE;

6)    IF  PREFIX  CODE  IS  14  OR  172;
```

Question 1 illustrates a prefix search in conjunction with a ranging (between or equal) search. All records would be located where CODE began with 11, 12, 13,,, 18 or 19.

Question 2 illustrates a prefix search and a non-prefix search combined. All employees with less than four dependents in either of the two departments would be located.

Question 3 illustrates a prefix search in conjunction with a ranging search with compound search values. All records would be selected where the first two digits of CODE were in either of the three indicated ranges.

Question 4 illustrates a prefix search on a non-numeric, non-hierarchical element. If a record had a SUBJECT entry of SHIP HULL DESIGN, the record would be selected in the search.

Question 5 illustrates a prefix search on a repeating element. Records where any two entries for SUBJECT began with WEAPONS and SUBMARINE would be selected. Thus a record with the two entries for SUBJECT -- WEAPONS SYSTEMS and SUBMARINE TECHNOLOGY -- would be selected.

Question 6 illustrates that compound search values with PREFIX may have different numbers of characters. All records where CODE began with either 14 or 172 would be selected.

5.1.4.10 Partial Search

Queries may be made on characters in the middle of a data element. This type of search is called a partial search. For example, assume that a date is entered into a file in the numeric form YYMMDD (YY-year, MM-month, DD-day). If the user wished to query on the month portion of the field, the query would be entered as follows:

```
IF PARTIAL DATE  IS   (2)12;
```

The query is interpreted as follows -- skip the first 2 characters of DATE and compare the next two characters on the value 12. Thus, all records would be selected with dates in the month December.

The general format for the partial search value is given below.

```
Field 1 & (NN) Field 2 & (NN) Field 3 ,,, Field N
```

Each field represents a contiguous set of characters in the data element to search on. The ampersand separates fields and NN represents the number of characters to skip between fields. The following queries illustrate the use of the partial search.

```
1)  IF PARTIAL CODE-A IS (3)AD;

2)  IF PARTIAL CODE-B IS AM&(2)PY;

3)  IF PARTIAL CODE-C IS (3)AP&(4)21;

4)  IF PARTIAL CODE-R BE (10)21/(10)35;
    AND PREFIX CODE-B IS AB?

5)  IF PARTIAL CODE-B IS
      (3) AD OR AP&(4)AX
    AND CODE-D IS AMPX?
```

In question 1, the 4th and 5th characters counting from the left are compared to the value AD.

In question 2, the 1st and 2nd characters are compared to the value AM and the 5th and 6th characters are compared to PY. Thus, the entry AMXRPYANR would satisfy the search.

In question 3, characters 4 and 5 are compared to AP and characters 10 and 11 are compared to 21. The value WXOAPUALU21456 would satisfy the search.

Question 4 illustrates a partial ranging search combined with a prefix search. If the 11th and 12th characters of CODE-R are in the range 21 to 35 and the first two characters of CODE-B are AB, the search is satisfied.

Question 5 illustrates a partial search with compound search values combined with a whole search. Note that the partial fields do not have to be the same character positions in the compound search values.

Spacing is optional in a partial search value on each side of the ampersand and on each side of the "skip" characters within the parentheses. Any spacing entered between the right parenthesis and the field value will be assumed to be part of the search value. For example, the following spacing is permitted.

```
( 3 )AP & ( 4 )21
```

5.1.4.11 Suffix Search

Queries may be made on the end (right-most) characters of a data element. This type of search is called a suffix search. The following examples illustrate.

```
1)  IF SUFFIX SUBJECT IS HULL DESIGN;
2)  IF SUFFIX CODE IS A6;
3)  IF PREFIX SUBJECT IS SHIP
    AND SUFFIX SUBJECT IS DESIGN;
```

In question 1, a record is selected if the data element SUBJECT has a value which ends in the words HULL DESIGN. For example, a SUBJECT entry of SHIP HULL DESIGN would satisfy the search.

In question 2, the search is satisfied if the data element CODE has a value which ends in the characters A6.

Question 3 illustrates a combination prefix and suffix search. In this case, a SUBJECT value which begins with the word SHIP and ends with the word DESIGN would satisfy the search. For example a SUBJECT entry of SHIP ENGINE DESIGN would satisfy the search.

In both Prefix and Suffix searches, the search is satisfied if the whole data value satisfies the search. For example, if the whole data value entered for SUBJECT is LOGISTICS, it would satisfy the search for all three of the following questions.

```
IF PREFIX SUBJECT IS LOGISTICS;
IF SUFFIX SUBJECT IS LOGISTICS;
IF SUBJECT IS LOGISTICS;
```

Both suffix and partial searches consume considerably more computer processing time than either prefix or whole value searches. For this reason, it is suggested that suffix and partial searches not be performed indiscriminately in interactive terminal runs since longer processing time and higher computer charges will result. It is suggested that these searches be performed in either batch terminal or off-line modes, particularly if the files being queried are large.

5.1.4.12 Efficiency Search

If a query is such that a user knows that the search will be satisfied by over 50 percent of the file, processing time can be decreased by performing an efficiency search. For example, assume a file contains the data element

TYPE which can have values A, B, or C; and over 98 percent of the data values are A. If a user wished to search the file for all records containing the value A, over 98 percent of the file would be selected. In this case, an efficiency search should be entered as shown below.

```
IF (E) TYPE IS A;
```

The E which indicates the efficiency search is always enclosed in parentheses. Spacing is optional on each side of the E.

While processing time can be decreased by performing an efficiency search, the processing time for this type of query will still be relatively large due to the large number of records which will be selected in the search (over 50 percent of a file). This should be considered before attempting this type of search in the interactive terminal mode.

5.1.4.13 Absent Search

Records may be located in a data base file which did not have data values entered for specified data elements. When records are added to a SHARP data base, only non-blank data values are entered. To locate records which had no entries for the data element CODE-A, the following query would be entered.

```
IF CODE-A IS ABSENT;
```

5.1.4.14 Present Search

The present search is the opposite of the absent search. This search allows all records to be located which have any values entered for specified data elements. To locate all records which have an entry for CODE-A, the following query would be entered.

IF CODE-A IS PRESENT;

Both the present and absent searches require a relatively large amount of computer processing time for large files and should not be performed indiscriminately in the interactive query mode.

5.1.4.15 Parallel (Linking Searches)

SHARP allows associator tags to be assigned to data values upon entering the data into a data base. These tags allow data to be mapped into arrays or to be associated with other data elements. For example, consider two data elements defined for a computer software data bank -- COMPUTER and OP/SYSTEM. COMPUTER is a repeating element which contains for each entry a computer on which the software is operational. OP/SYS is a repeating element which contains for each entry the corresponding operating system used. The table below illustrates the entries for a typical record in the data base.

COMPUTER	TAG (LINK)	OPERATING SYSTEM
GE 635	A	GECOS
UNIVAC 70/45	B	DOS
CDC 6700	C	SCOPE
IBM 360	D	OS

Corresponding entries from the two arrays are related. Thus, the software is operational on the CDC 6700 under the SCOPE operating system. To associate the two data arrays, a linking tag is assigned. In this case, consecutive tags beginning with A are assigned. Corresponding information is in the same LINK. When multiple arrays contain related information, they are said to be parallel.

To query on corresponding information from parallel arrays, a link search is performed.

Example:

```
IF COMPUTER IS IBM 360 (LINK)
AND OP/SYS  IS  OS (LINK);
```

The above question is interpreted as follows -- locate all records where the computer is IBM 360 and the operating system is OS in the same link. The above sample record would satisfy the question, since it contains the stated values in the same link (D).

Example:

```
IF COMPUTER IS IBM 360 (LINK)
AND OP/SYS IS  DOS (LINK);
```

The above question would not be satisfied by the record since it does not contain the data values in the same link. (IBM 360 is in link D, DOS is in link B).

A non link search may be performed on the two arrays.

Example:

```
IF COMPUTER IS IBM 360
AND OP/SYS  IS  DOS;
```

This question is interpreted as follows -- Locate all records where the computer array contains IBM 360 and the OP/SYS array contains DOS regardless of which link either value is in. In this case the user might be interested in whether the software is on the 360 regardless of operating system and whether it is under DOS on any computer.

Questions may be entered with both linked and non linked search values.

Example:

```
IF COMPUTER IS IBM 360 (LINK)
AND OP/SYS IS OS (LINK)
AND APPLICATION IS SHIP DESIGN;
```

In this case, computer software would be selected which is operational on the IBM 360 under OS and is used for ship design applications.

Linking searches may be performed on data elements which are not repeating. For example, consider an accounting file with cost information COST1 and COST2. Assume that each data element is not repeating (only one data value entered per record) and that each cost value entered is associated with one of the four quarters of the year. The user may assign a link tag to represent the associated quarter. (A = 1st quarter, B = 2nd quarter, C = 3rd quarter, D = 4th quarter). The following table illustrates:

RECORD NO.	COST 1/QTR.	COST 2/QTR.
1	2500, A	5000, A
2	4000, A	3200, B
3	4000, C	3500, C
4	5000, D	1250, A

If a user wished to locate all records, where COST1 and COST2 exceeded 3000 in the same quarter, the following query could be entered.

```
IF COST1 GR 3000 (LINK)
AND COST2 GR 3000 (LINK);
```

In the above example, only record 3 satisfies the search. In it, both costs exceed 3000 for the third quarter. Note that record 2 does not satisfy the search, since both costs are not associated with the same quarter.

5.1.4.16 Specific LINK/ROLE Searches

Queries may be made on specific associator tags assigned to data values. Either or both of the associator tags may be specified in a query. In the previous example in Section 5.1.4.15, if the user desired all records where COST1 exceeded 3500 in quarter 1 and COST2 exceeded 3000 in quarter 2, the following query would be entered.

```
IF COST1 GR 3500(LINK = A)
AND COST2 GR 3000(LINK = B);
```

Record 2 from the previous example would satisfy this search.

The following questions illustrate various examples of searches where associator tags are specified.

```
1)  IF CODEA IS 120(LINK = A)
    AND CODEB IS 50(ROLE = 25);

2)  IF CODEA IS 50(LINK = A, ROLE = 4)
    AND CODEB IS 75(ROLE = 10, LINK = C);

3)  IF CODEA IS 25(LINK = B)
    AND CODEB IS 50(ROLE = 7, LINK = R)
    AND TOTAL IS LT 12;

4)  IF CODEA IS A45(LINK = M) OR
       A51(LINK = R, ROLE = 87);
    AND CODED IS 50(LINK = A) AND 60(LINK = R);
```

In question 1, the Link is specified for the first search value and the role is specified for the second. A record to be selected must contain CODEA with a value of 120 in Link A and CODEB with a value of 50 in Role 25.

Question 2 illustrates the specification of both associator tags for each search value. This type of search allows users to search two dimensional arrays by array location, each location being identified by the corresponding associator tags. Note that when both LINK and ROLE are specified for a search value, either may be specified first.

Question 3 illustrates a LINK/ROLE search combined with a non LINK/ROLE search. For a record to satisfy the search, all three data elements must contain values which satisfy the specified restrictions.

Question 4 illustrates a search with compound search values. Note that CODED must be a repeating element for the search to be meaningful.

In specifying LINK/ROLE searches, spacing is optional - between the search value and the left parenthesis, between the left parenthesis and the word LINK or ROLE, on each side of the equals sign, on each side of the comma, and before the right parenthesis. The equals sign may be optionally omitted, in which case at least one space must be used in place of the equals sign. Leading zeroes may be entered or omitted (ROLE = 4, ROLE = 04). In general, spacing rules need not be memorized. If the information is entered as it would naturally be written, the spacing will be correct.

5.1.4.17 Parallel Searches with Specific Roles

Parallel searches may be made with specific Role tags specified.

Example:

```
IF N IS 140(LINK, ROLE = 2)
AND P IS 35(LINK, ROLE = 5);
```

The above search is interpreted as follows:

Locate all records where N has a value of 140 and P has a value of 35 with the same associated link. In addition the specified value for N (140) must occur with an associated role tag of 2 and the value of P must occur with an associated role tag of 5. The following table illustrates the selection process:

RECORD	N(Tags)	P(Tags)	Record Selected
1	120(A,4)	30(A,5)	NO
	140(B,2)	35(B,6)	
	170(C,12)	190(C,15)	
2	110(A,1)	15(A,32)	YES
	130(B,10)	17(B,45)	
	140(E,2)	35(E,5)	
	170(F,17)	42(H,6)	
3	130(A,16)	10(B,21)	YES
	140(C,2)	35(C,5)	

Record 2 is selected because both data values occur in the same link (E) with the specified roles. Record 3 is selected with the data values occuring in a different link (C). Record 1 is not selected because although the data values occur in the same link (B), the value for P is not in the designated role.

Parallel searches can be performed only on the Link value. The following examples illustrate.

```
Correct          IF CODE A IS 10 (LINK)
                 AND CODE B IS 20 (LINK)

Correct          IF CODE A IS 10 (LINK, ROLE 20)
                 AND CODE B IS 15 (LINK, ROLE 30);

Correct          IF CODE A IS 10 (ROLE = 12);

Incorrect        IF CODE A IS 15 (ROLE)
                 AND CODE B IS 20 (ROLE);

Incorrect        IF CODE A IS 12 (ROLE, LINK = A)
                 AND CODE B IS 15 (ROLE, LINK = C);

Correct          IF CODE A IS 10 (LINK = A, ROLE = 16)
                 AND CODE B IS 20 (LINK = C, ROLE = 50);
```

5.2 Sort Specification

With each query question entered, users may specify the desired sort sequence of the output records selected during the search. Any combination of those data elements defined in the User's Data Base Definition may be specified for the sort sequence. The data elements may be sorted on in both ascending (low to high values) and descending (high to low values) sequence. The data elements selected for sorting may contain up to 200 characters of information. If a question does not contain a sort specification, the records selected for that question will be output sequenced by record key in ascending order.

Examples:

```
QUERY JJS
IF KEYWORD IS SYSTEMS ANALYSIS
   OR COMPUTER PROGRAMMING;
 SORT ON AUTHOR.

IF KEYWORD IS NUMERICAL CONTROL;
SORT BY HIGH DATE AND LOW AUTHOR

IF PREFIX KEYWORD IS LOGISTIC;
SORT ON ORG/NAME, AUTHOR, AND
 HIGH DATE.

IF PREFIX KEYWORD IS LOGISTIC;
THEN SORT ON ORG/NAME AND AUTHOR
 AND HIGH DATE.
 $END
```

In question 1, the records selected are to be output sequenced by Author in ascending order. Ascending order (low to high) is the default sequence. For example, if the search produced four hit records with authors Smith, Jones, Brown, and Williams, the output records would be sequenced as follows -- Brown, Jones, Smith, Williams. Note that author and any other data element which is a personal name should always be entered into a data base last name first (left-most characters), since the last name is the most important part of the name for both query and sort purposes.

In question 2, the output records are to be sequenced by two data elements -- date (high to low) and author (low to high). In this case, the user is interested in records with most recent dates output first, and those records with the same date are to be sequenced by author in ascending order. Date values would have been entered into the data base with the most important part (year) entered left adjusted in the field in numeric format (i.e., Feb 17, 1975, would be entered 750217). The example below illustrates:

Records Selected (File Sequence)

Date	Author
740312	SMITH A.P.
740415	WILSON N.M.
750121	JONES J.R.
740415	ADAMS B.V.
750121	BROWN A.C.
750510	SMITH R.N.

Records Output (Sort Sequence)

Date	Author
750510	SMITH R.N.
750121	BROWN A.C.
750121	JONES J.R.
740415	ADAMS B.V.
740415	WILSON N.M.
740312	SMITH A.P.

Note that the LOW modifier was used in question 2 before AUTHOR to avoid ambiguity. To specify -- SORT BY HIGH DATE AND AUTHOR -- is somewhat ambiguous, since the user may intend the HIGH modifier to apply to both DATE and AUTHOR. Whenever the statement could be ambiguous, both modifiers should be used.

In question 3, the sort sequence is formed from three data elements -- ORG/NAME (ascending), AUTHOR (ascending), and DATE (descending). Note that it is not ambiguous to leave out the LOW modifiers for ORG/NAME and AUTHOR, since LOW is automatically in effect until changed to HIGH for DATE. The commas and conjunction AND allow the statement to be entered exactly as it would be written in English.

Question 4 is interpreted the same as question 3. A different grammatical structure is illustrated.

The acronyms used to specify the data elements desired for sorting must be entered exactly as defined in Data Base Definition. The general syntax for the normal sort specification is given below (underlined fields are non essential and are allowed for readability).

```
THEN SORT   ON    LOW    NAME1  ,  AND
            BY    HIGH

            LOW          NAME2  ,  AND
            HIGH

            LOW
            HIGH         NAME3  ,  AND

             .            .     .   .
             .            .     .   .

            LOW
            HIGH         NAMEN  .
```

Note that a period at the end of the statement is optional. The allowable spacing between fields for the sort statement is the same as that for all SHARP free form syntax (at least one space between fields unless separated by a comma. Spacing is optional on each side of commas and period).

5.2.1 Prefix Sort Specification

This specification allows a user to sort on the beginning (left-most) characters of a data element value.

Examples:

1) SORT ON (2)DATE.

2) SORT ON HIGH (2)DATE AND LOW AUTHOR.

3) SORT ON (20)ORG/NAME.

In example 1, the sort is to be only on the first two (left-most) characters of the field DATE (the year part in the format YYMMDD).

In example 2, the sort is to be on two fields -- the first two characters of the date (high to low sequence) and author (low to high sequence).

In example 3, the sort is to be on the first 20 characters of ORG/NAME.

Spacing is optional between the right parenthesis and the acronym and within the parentheses on each side of the number.

The prefix sort allows sorting on the first part of hierarchical fields or fields in which the first part is the important one, such as personal names with last name first.

A maximum of 90 characters is allowed for prefix sorting (number within parentheses).

5.2.2 Partial Sort Specification

This specification allows users to sort on characters in the middle of a data element value.

Examples:

```
1)    SORT ON (*2,2)DATE.
2)    SORT ON HIGH (*2,2)DATE.
3)    SORT ON (*6,20)NAME.
```

In example 1, the sort is to be on the third and fourth characters of DATE (skip the first two characters and sort on next two characters -- the month part of DATE in the format YYMMDD).

In example 2, the sort is the same except it is to be high to low (month 12-Dec to month 1-Jan).

In example 3, the sort is to skip the first six characters of name and sort on the next 20 (characters 7 through 26).

This type of sort is often used for fields which have some type of extraneous information appended to the fields.

5.3 Print Specification

Users may enter with each query question a print statement which specifies the data elements to be output from each record selected in the search. The data elements are identified via their acronyms assigned in Data Base Definition. If a print statement is entered without a report statement, the data elements selected will be output in a standard system format. If a report statement is also entered, the information will be output in the format specified when the report was defined (see Section 5.4 for a complete description of the report statement).

Examples:

```
QUERY DPC
IF QTY GR THAN 500;
PRINT UIC, UNITP, AND ALLOW.
SORT ON UIC

IF QTY BE  500/565;
PRINT ALL.
SORT ON UIC AND FSN.

IF EIC IS  N810000;
SORT ON HIGH DATE.
PRINT SHIP AND DATE AND ETOR
    AND EQUIPNOM.
$END
```

In question 1, three data elements -- UIC, UNITP, and ALLOW are to be retrieved from each record which satisfied the search and output in a standard system format sequenced by UIC.

In question 2, all data elements present in each record selected will be retrieved and output. ALL is a keyword and therefore must be entered exactly as shown. Also, ALL may not be chosen as a data element acronym in Data Base Definition, since ambiguity would result.

In question 3, the four data elements specified are to be output in standard format sequenced by date (high to low).

5.3.1 Standard System Print Format

In the standard system print format, if the information specified in the print statement can be output on a single line, a columnar report is formatted and output. If the information cannot be output on a single line, a linear type report is formatted with multiple lines of output appearing for each record selected in the search. The system will examine the SIZE and MAXSIZE fields in Data Base Definition to determine the maximum number of print characters required to output the data elements specified. This number is compared to the number of characters which can be put on a print line (72 for on line reports, 132 for off line reports).

For columnar outputs, each line across the page will represent a record selected in the search. At the top of the page will appear any variable heading information if entered with the print statement (see Sec. 5.5 "Variable Headings" for a complete description of how variable headings are entered). Under the variable headings will appear the acronyms of the data elements to be printed. The acronyms will comprise column headings for the report.

Example:

```
              RECORDS WITH QTY GR THAN 500
                        QSR FILE

DPC001                                            JUN 10 1975

  UIC                 UNITP               ALLOW

  54040                143                   1

  54049                 22                  20

  54049                 49                  55
```

The above report could be in response to the print statement in question 1 of the example in section 5.3. The two lines at the top are hypothetical variable headings which could have been entered with the question. The next line of information contains the query identification, the question number within the query (001), and the current date. The acronyms then appear over each column of information. The data elements will appear in the order specified. When a print page is filled (approximately 11 inches), a new page is begun with the headings repeating.

For linear outputs, the data values for each data element is output preceded by its acronym. The following example illustrates:

```
                        CASREP FILE
                  MONITOR AIRPAC CASREPS

DPC002                                            JUN 10 1975

SHIP:  N65   DATE:  JUN 28  72   DTG: 28184862
RDCODE:  3   EQUIPNOM:  AN/WLR-ID   JULIAN:  2213
ETOR:  JUL 30 72   SRDTG:   15180572   EIC: N810000 PMS: Y

SHIP  N65   DATE:  JUN 25 72    DTG:  25213962
RDCODE: 2   EQUIPNOM:  AN/SPN-42    JULIAN:  2187
ETOR:  JUL 23 72   SRDTG:   15190572   EIC: PDØE000 PMS: Y
```

The above information illustrates output from two records selected in the search. The print statement which caused the generated output would be:

```
        PRINT  SHIP, DATE, DTG,  RDCODE,
             EQUIPNOM, JULIAN, ETOR,
              SRDTG,  EIC, AND  PMS.
```

5.3.2 Prefix Print Specification

This specification allows a user to print only the beginning (left-most) characters of a data element. It is analagous to the prefix sort statement and has exactly the same format.

Examples:

```
QUERY TM
IF KEYWORD IS LOGISTICS;
SORT ON (2)DATE.
PRINT (2)DATE

IF KEYWORD IS LOGISTICS;
SORT ON HIGH (2)DATE AND LOW
   AUTHOR.
PRINT  (2)DATE AND AUTHOR.
$END
```

In question 1, only the first two characters of DATE are to be output. In question 2, the first two characters of DATE and the whole AUTHOR name is to be output.

5.3.3 Partial Print Specification

This specification allows users to print data characters in the middle of a data element value. It is analagous to the partial sort specification and the format is exactly the same.

Examples:

```
1)     PRINT     (* 2,2) DATE.

2)     PRINT     (* 6,20) NAME.
```

In example 1, the third and fourth characters of DATE are output (the month part of the date in the format YYMMDD).

In example 2, the first six characters of NAME are skipped and the next 20 are printed (characters 7 through 26).

5.3.4 Print Computation Specification

This specification allows computations to be performed on data elements in those records selected in the search. The following computation specifications are allowed:

Computation	Acronym(s)	
Sum or total a data element	SUM TOTAL	
Average of a data element	AVG AVERAGE	
Percent (1 element ÷ another x 100)	PERCENT	
Percent sum (sum two elements then compute percentage)	PERCENT	SUM TOTAL
Maximum of a data element	MAXIMUM MAX	
Minimum of a data element	MINIMUM MIN	
Standard deviation of a data element	SIGMA	
Count the occurrences of a data element	COUNT	
Sub-total a data element when another element changes in value	SUBTOTAL SUBTOTALS SUB-TOTAL	

The PERCENT computation allows the percentage to be computed from two specified data elements in a record. PERCENT is the only computation listed above where a computed value is output with each hit record. For the other computations, a single computed value is output for all the hit records selected. For example, if TOTAL is specified for a data element, and 50 records are selected in the search, the values for that data element in the 50 records are added together and a single sum is computed.

The computation PERCENT SUM or PERCENT TOTAL allows two data elements to be specified. The data values for each element are summed separately, and then the sum of the first data element is divided by the sum of the second. This quotient is then multiplied by 100 to form the single percentage value. The following examples illustrate computation specifications.

```
1)          IF QTY1 GR 50;
            SORT ON FSN.
            PRINT QTY1 AND QTY2
            PRINT SUM QTY1 AND QTY2

2)          IF QTY1 GR 50;
            SORT ON FSN.
            PRINT QTY1 AND QTY2
            PRINT SUM QTY1 AND SUM QTY2

3)          IF QTY1 GR 50;
            SORT ON FSN.
            PRINT QTY1, QTY2, AND
               SUM QTY1 AND QTY2

4)          IF QTY1 GR 50;
            PRINT QTY1 AND SUM QTY2.
            SORT ON FSN

5)          IF QTY1 GR THAN 50;
            PRINT QTY1, QTY2, AND QTY3.
            PRINT TOTAL QTY1 AND QTY2
               AND MAXIMUM QTY1 AND
               MINIMUM QTY2.
            PRINT SIGMA QTY3 AND
               COUNT QTY4.  SORT ON FSN.
            PRINT SUB-TOTAL QTY3 ON FSN.
```

```
6)          IF QTY1 BETWEEN  50/150;
            SORT ON FSN.
            PRINT QTY3 AND QTY4 AND
              SUBTOTAL QTY3 AND QTY4 ON FSN.

7)          IF QTY1 GR 75;
            SORT ON FSN AND UIC.
            PRINT QTY1, QTY2, QTY3 AND FSN.
            PRINT SUBTOTAL QTY1 AND SUBTOTAL
              QTY2 ON FSN.
            PRINT MAX QTY1 AND MIN QTY2.
            PRINT AVG QT3.

8)          IF QTY1  GR  75;
            PRINT QTY1, QTY2, AND QTY3.
            PRINT PERCENT QTY1/QTY2.

9)          IF QTY1  GR  75;
            PRINT QTY1, QTY2, AND QTY3.
            PRINT PERCENT  QTY1/QTY2
              AND QTY2/QTY3.

10)         IF QTY1  GR  75  AND UNITP
              LESS THAN  340;
            PRINT QTY1, QTY2, QTY3, AND QTY4.
            PRINT PERCENT SUM QTY1/QTY2
              AND QTY3/QTY4
```

In question 1, QTY1 and QTY2 are to be output and summed. Following the keyword SUM, any number of data elements may be entered which are to be summed. The SUM computation stays in effect until a different keyword is encountered.

Question 2 illustrates that the keyword SUM may be optionally entered before each data element to be summed.

Question 3 illustrates that the computation specification does not have to be entered in a print statement separate from the one used to specify the data elements to be output.

In question 4, the sum of QTY2 is to be output but QTY2 is not to be printed. Thus, a computation may be output without outputting the data values which were used in the computation. Question 4 also illustrates that either the PRINT or SORT statement may be entered first.

Question 5 illustrates various combinations of computation specifications. All of the allowable computations may be specified with a question. Note that TOTAL and SUM are the same computation, and that two or more different computations may be specified in the same print statement. The SUB-TOTAL specification indicates that the data values for QTY3 in the records selected are to be subtotaled whenever FSN changes in value from one record to the next. Normally a subtotal is meaningful only if the change element (FSN in this case) is also the sort key. An exception would be the case where subtotals were desired with each output record. This can occur when a user wishes to total the entries in an array within a record. In this case, the change element would be the record key regardless of the sort key. The COUNT specification will count the number of times that QTY4 had a data value entered in the records selected. The SIGMA specification will cause the standard deviation to be computed of all data values for QTY3. The MAXIMUM and MINIMUM specifications will cause the largest and smallest values, respectively, to be output for QTY1 and QTY2 in the records selected.

In question 6, a different syntax (SUBTOTAL) is used for specifying subtotals.

In question 7, various combinations of allowable syntax are illustrated.

Question 8 illustrates the PERCENT computation. A percentage is to be computed for each record selected by dividing QTY1 by QTY2 and multiplying the result by 100. Spacing is optional on each side of the slash.

Question 9 illustrates a percentage computation on two pairs of data elements.

Question 10 illustrates the PERCENT SUM computation. For the first percentage, two sums will be computed -- the sum of all QTY1 values in the selected records and the sum of all QTY2 values. Then, the percentage is computed by dividing the QTY1 sum by the QTY2 sum and multiplying the quotient by 100. The sum percentage is similarly computed for QTY3/QTY4.

The following example illustrates the format in which computations are output in the standard system print format.

```
QUERY LM
IF ALLOW GR 0;
PRINT UIC, UNITP, AND ALLOW.
PRINT SUM UNITP AND ALLOW.
PRINT MAX UNITP AND MIN ALLOW.
PRINT AVG UNITP AND ALLOW.
SORT ON UIC

IF PMS IS Y; PRINT SHIP, DATE,
DTG, RDCODE, EQUIPNOM AND JULIAN.
SORT ON SHIP.  PRINT MAX RDCODE.
$END
```

```
                       (SAMPLE OUTPUT)


LM  001                                              JUN 10 1975


 UIC                      UNITP                      ALLOW

54048                      143                          1
54049                       22                         20
54049                       49                         55


                    * * * TOTAL * * *
                           214                         55

                    * * * MAXIMUM * * *
                           143

                    * * * MINIMUM * * *
                                                        1

                    * * * AVERAGE * * *
                            71                         25



LM 002                                               JUN 10 1975


SHIP: N64  DATE: JUN 28 72   DTG: 28184862
RDCODE: 3  EQUIPNOM:  AN/WLR-1D   JULIAN: 2213

SHIP: N65  DATE: JUN 25 72   DTG: 25213962
RDCODE: 2  EQUIPNOM:  AN/SPN-42   JULIAN: 2187


 * * * MAXIMUM * * *
 RDCODE: 3
```

For the first question, the three data elements are printed on a single line forming a columnar type report. The computations are output below their respective columns.

For the second question, the data elements will not fit on a single line and are output on two lines forming a row type report. The requested computation is output at the bottom of the report. The field on which the computation is made is identified by its acronym.

The following computations should be performed only on data elements defined as INTEGER or REAL in Data Base Definition -- TOTAL, SUBTOTAL, PERCENT, PERCENT SUM, AVERAGE, and SIGMA.

MAXIMUM, MINIMUM, and COUNT may be performed on other data element formats.

The SUBTOTAL specification always requires that a change variable be specified. A change variable may optionally be specified with the other computations. If used, the computation will be performed whenever the specified variable changes in value.

Example:

```
PRINT  MAX  QTY1 AND QTY2 ON FSN.
PRINT  SIGMA  QTY2, QTY3 AND QTY4 ON FSN.
PRINT AVG  QTY2 ON UIC.
```

Note that a change variable is always introduced by the keyword ON.

The prefix or partial capability may also be used with change variables in conjunction with a computation.

Examples:

```
QUERY RW
IF SERIES IS 0334, 1515, OR 1520
AND SALARY GR THAN 9000;
SORT ON SERIES AND HIGH GRADE.
PRINT SERIES, GRADE, AND SALARY.
PRINT AVERAGE GRADE AND SALARY
   ON SERIES.

IF JOB-TITLE IS COMPUTER SPECIALIST
   OR COMPUTER PROGRAMMER:
PRINT LOC-CODE, JOB-TITLE, SALARY,
   GRADE, AND NAME.
SORT ON LOC-CODE AND JOB-TITLE.
PRINT COUNT NAME ON (3)LOC-CODE.
PRINT AVG GRADE ON (3)LOC-CODE.

IF SALARY GR 15000;
SORT ON (*2,2)E-DATE.
PRINT COUNT NAME ON (*2,2)E-DATE
$END
```

In question 1, the output is sequenced by SERIES and GRADE. The average GRADE and SALARY will be output for each different SERIES.

In question 2, LOC-CODE could be a hierarchical code where the first 3 characters represent division number. The number of persons and the average grade for each division would be output.

In question 3, the output is sequenced by the third and fourth characters of E-DATE, which could represent the month portion of the date. The number of persons for each month would then be computed.

5.3.5 Parallel Print Specification

The parallel print specification allows information to be printed which has the same Link associator tag as one or more data elements in the associated question.

Examples:

```
1)     IF COMPUTER IS IBM 360 (LINK);
       PRINT OP/SYS (LINK).

2)     IF COMPUTER IS IBM 360 (LINK)
       AND OP/SYS IS OS (LINK);
       PRINT LANGUAGE (LINK).
         AND APPLICATN.

3)     IF COMPUTER IS IBM 360 (LINK)
       AND APPLICATN IS SHIP DESIGN;
       PRINT LANGUAGE (LINK)
         AND CONTACT.

4)     IF COMPUTER IS CDC 6700 (LINK)
       AND OP/SYS IS SCOPE (LINK)
       AND APPLICATN IS PERSONNEL;
       PRINT LANGUAGE (LINK) AND
         CONTACT

5)     IF COMPUTER IS IBM 360;
       PRINT  OP/SYS.
```

The above questions illustrate searches on the Computer Software Data Bank described in Section 5.1.4.15. COMPUTER, OP/SYS, and LANGUAGE are parallel arrays whose corresponding entries are related.

In question 1, all records are to be located where the computer array contains IBM 360. The corresponding entry in the OP/SYS array is to be output. Thus, if the software is operational on the IBM 360, the IBM 360 Operating System is to be output.

Question 2 illustrates a parallel search in conjunction with a parallel print. If the software is operational on the IBM 360 under the OS Operating System, the corresponding language used in developing the software is output along with the application (APPLICATN). Note that APPLICATN is a nonparallel, nonrepeating element. Thus, the software may be operational on different computers, under different operating systems, and written in different languages; but there is only one application.

Question 3 illustrates a nonparallel search in conjunction with a parallel print. In this case, all records will be selected where the software is on the IBM 360 and its application is Ship Design. The language used for the software on the IBM 360 is output along with the contact (designated person) for the software.

Question 4 illustrates a parallel search/print combined with a nonparallel search/print.

In question 5, no parallelism is specified. In this case, if the software is on the IBM 360, all operating systems it is under (on the IBM 360 and all other computers) will be output. If an array is specified in a print statement and no parallelism is indicated, all entries in the array will be output.

Parallel printing may be performed on data elements other than arrays. In the accounting example given in Section 5.1.4.15, two data elements COST1 and two COST2 have link tags assigned to indicate quarter of the year.

(A = 1st quarter, B = 2nd quarter, etc.). Assume a user wished to locate all records where COST1 was greater than 2500 in any quarter and to output COST2, if it has an entry for the same quarter, the following query illustrates:

```
IF COST1 GR 2500 (LINK)
   PRINT COST2 (LINK)
```

Note that a record could satisfy the search but not have a COST2 for the same quarter. In this case, no cost data would be output for the record.

In parallel searches on arrays, as illustrated by the software data bank, multiple entries may satisfy the search. In this case, if a parallel print is specified, there will be one output data value for each array entry which satisfied the search.

Example:

```
IF COMPUTER IS IBM 360 (LINK);
  PRINT  OP/SYS (LINK).
```

In this example, assume that a record contained the following entries in the computer and operating system arrays:

Computer	Operating System
GE 635	GECOS
UNIVAC 70/45	DOS
CDC 6700	SCOPE
IBM 360	OS
IBM 360	DOS

In this case, two entries in the computer array satisfies the search. Thus, the two corresponding entries in the operating system array (OS and DOS) will be output by the print statement.

5.3.5.1 Parallel Print with Specific Roles

In 2 dimensional arrays or other situations where both associator tags are used, the specific role tag may be indicated in the print statement.

Examples:

```
1)      IF CODEA IS 20 (LINK);
        PRINT CODEB (LINK, ROLE=12).

2)      IF CODEA IS 20 (LINK, ROLE=16);
        PRINT CODEB (LINK, ROLE=15).

3)      IF CODEA IS 20 (LINK, ROLE=16)
        AND CODEB IS 12 (LINK, ROLE=21);
        PRINT CODED (LINK, ROLE=26).
```

In question 1, a parallel print is specified. However, in addition to having the same link as CODEA, CODEB output values must have an assigned role of 12.

Question 2 is similar except the role is specified in the search.

In question 3, a parallel search is performed on CODEA and CODEB with the specified roles. Corresponding values of CODED are output with the specified role of 26.

5.3.5.2 Print Statement with Specific Link/Roles

The specific link and/or role associator tags may be indicated in a nonparallel print statement.

Example:

```
IF CODEA IS 10;
PRINT CODEB (LINK=A, ROLE=12)
  AND CODED (LINK=A)
    AND CODEF (ROLE=15)
```

In this case only CODEB values having the specified Link and Role will be output. Only CODED values in Link A will be output and only CODEF values in role 15 will be output.

5.3.6 Parallel Sort Specification

A user may specify a parallel sort, which is analagous to a parallel print.

Example:

```
IF CODEA IS 10(LINK);
PRINT CODEB (LINK) AND CODEC.
SORT ON CODEB (LINK).
```

In this example, the values of CODEB which are printed in the parallel print are used for the sort.

Example:

```
IF CODEA IS 10 (LINK)
AND CODEB IS 20 (LINK);
PRINT CODEC (LINK) AND CODED (LINK)
SORT ON CODEC (LINK) AND CODED (LINK)
```

This example illustrates multiple elements in the parallel sort.

5.3.6.1 Parallel Sort With Specific Roles

A parallel sort may be indicated with a specific role tag.

Example:

```
IF CODEA IS 10 (LINK);
PRINT CODEB (LINK, ROLE=10) AND CODEC
SORT ON CODEB (LINK, ROLE=10)
```

5.3.6.2 Sort Statement With Specific Link/Roles

The specific link and/or role associator tags may be indicated in a nonparallel sort statement.

Examples:

```
IF CODEA IS 10;
PRINT CODEB (LINK=A) AND CODEF.
SORT ON CODEB (LINK=A)

IF CODEA IS 10;
PRINT CODEG, CODEF, AND CODEM.
SORT ON CODEP (ROLE=10)

IF CODEB IS 20;
PRINT CODEG (LINK=B, ROLE=21),
  CODEF, AND CODEM.
SORT ON CODEG (LINK=B, ROLE=21).
```

In the above questions, the sort will be performed only on the element values which have the specified links and roles.

5.4 Report Specification

Users may enter with each query question a report statement which identifies a previously defined report. Records selected in the search will then be output in the format which was specified when the report was defined. (See Section 6, "Report Definition" for a complete description of report formats available.) If no report statement is entered, the selected information will be output in the standard system print format described in Section 5.3.1. The following examples illustrate the report specification.

```
QUERY HPS
IF QTY GR THAN 500;
SORT ON UIC.
REPORT QSR

IF QTY GR THAN 500;
SORT ON FSN.
PRINT UIC, UNITP, AND ALLOW.
REPORT QSR.

IF SHIP IS N65;
REPORT C1
SORT ON EIC
PRINT SHIP, DATE, RDCODE,
EQUIPNOM, ETOR, AND (3)DTG.
$END
```

In question 1, the records selected in the search are to be output in a report format which was catalogued under the name QSR when the report was defined. The report statement is introduced by the keyword REPORT. The report identification (QSR) immediately follows. A period may be optionally entered following the report name. A different report may be specified with each question.

In question 2, a print statement is entered with a report statement. When report QSR was defined, data elements in addition to the three indicated in the print statement may have been specified for inclusion in the report. The print statement will cause only the three data elements specified to be output in the format defined by report QSR. Thus, the print statement can be used to select only a subset of the original data elements defined for output in the report definition. When no print statement is entered, all the data elements specified in report definition will be output.

Question 3 is similar except only the first three (left-most) characters of element DTG are to be output in report C1 format. The whole data values of the other elements are to be output in the report. The prefix and partial print capabilities may be used to further limit the output report information by screening out parts of the data element values. This may be desirable in the case of data elements which represent hierarchical structures.

Computations may be specified at query time in conjunction with a defined report. For example, a user may define and catalog a report. Then, during inputting of a query, he may decide to select a report and perform computations not specified when the report was defined. The following examples illustrate:

```
QUERY SW
IF QTY GR THAN 500;
SORT ON FSN
REPORT QSR.
PRINT SUM UNITP AND ALLOW.
PRINT SIGMA UNITP

IF QTY1 GR THAN 500;
SORT ON FSN
PRINT UIC, UNITP, AND ALLOW.
PRINT AVERAGE QTY1
REPORT QSR
$END
```

In the first question, the selected records are to be output in the format defined for report QSR. Data elements UNITP and ALLOW are to be totaled and their sums are to be output with the report. In addition, the standard deviation is to be computed and output for UNITP. Computations may be output on data elements which do not appear in each output report record.

In the second question, only the data elements UIC, UNITP, and ALLOW are to be output in report format QSR. The average is to be computed and output for element QTY1 which is not put out in each output report record.

5.5 Variable Heading Specification

Users may enter with each query question from one to three lines of information which is to appear at the top of each page of the output report. If a report specification is also entered, the variable headings will appear on output directly under the master heading at the top of each page. If a report is not requested, the variable headings will appear on output at the top of each page. The variable headings will be centered to the output page width.

Examples:

```
QUERY BV
IF ACTVTY IS CFY?
SORT ON START DATE.
REPORT A
HEAD 1 'SHIPS WITH ACTIVITY OF CFY
        SEQUENCED BY START DATE'


IF PYRYRFND GR 2500 AND ACTVTY IS CFY;
REPORT B
SORT ON SHIPID
HEAD 1 'SHIPS WITH ACTIVITY CFY AND'
HEAD 2 'PRIOR FUNDING GR THAN 2500'
HEAD 3 'SEQUENCED BY SHIPID'


IF CODEA IS A21 OR A25;
PRINT CODEA, CODEB, CODED, AND
   DATE.
REPORT C5.  SORT ON HIGH DATE.
HEADING 1 $RECORDS WITH CODEA$
HEAD 2 $A21 OR A25 - JOHN JONES$
$END
```

The keyword HEAD introduces each variable heading line. Each heading line may have from one to 60 characters of information. In question 1, one heading line is specified. The heading line is bounded by single quotation marks. A dollar sign may optionally be used in place of the single quotes. The heading line number immediately follows the keyword HEAD or HEADING.

In question 2, three variable headings are entered. Each heading line entered will appear as one output heading line in the printed report.

In question 3, two variable headings are entered. Each heading line is bounded by a single dollar sign.

Variable headings are optional and are normally used to annotate the output report with information which describes the selection criteria used in the question, the sort sequence, the user name or any other pertinent information. In questions 1 and 2, the headings describe the criteria by which the records will be selected and the sort sequence. In question 3, the selection criteria is described and a personal name is entered.

The SHARP DMS places no content restrictions on the information which is entered into variable headings. However, users of a particular data bank may be required by the data bank sponsors to enter standard information into the variable headings. It is recommended for ad hoc queries, that information describing the selection criteria be entered.

5.6 Input Query Error Correction

5.6.1 Correction of Syntax or Content Errors

During an interactive query, error diagnostics are output immediately to the user as they occur. The user may then correct his errors interactively.

Example:

```
QUERY DC
IF SERIS IS 0334;

FATAL ERROR--INVALID SEARCH
ACRONYM---QUERY DC, QUESTION 001
THE TERM SERIS WAS NOT FOUND
IN USER TERM TABLE.  ABOVE QUESTION
WAS NOT PROCESSED.  PLEASE EXAMINE
FOR SYNTAX ERROR AND CONTENT RE-ENTER
IF POSSIBLE.

IF SERIES IS 0334;
PRINT NAME AND JOB-TITLE
$END
```

In this example, the search acronym SERIES is initially incorrectly spelled. The user is informed of this via the interactive print out from the system. The user then re-enters the question with the acronym spelled properly.

Section 9, "SHARP ERROR MESSAGES", lists all the error diagnostics which may occur during the interactive inputting of a query.

5.6.2 Deletion of Questions ($DELETE)

A user may desire to delete a question which has been entered but which does not contain any syntax or content errors which would cause error diagnostics to be output. An entire question consisting of one or more input lines may be deleted by using the $DELETE command.

Examples:

```
QUERY CEP
IF SERIES IS 0334
AND SALARY GR THAN 15000?
$DELETE

IF SERIES IS 1515
AND SALARY GR THAN 15000?
PRINT SSN, NAME, AND SKILL.

IF SKILL IS COMPUTERS $DELETE

IF SKILL IS COBOL PROGRAMMING?
PRINT NAME, SSN, AND SALARY.
$END
```

In the first question in the above examples, the user has entered the wrong search value for SERIES. He observed the error after the question had been entered. The $DELETE command is entered which will delete the entire question. The question is then re-entered with the proper search value for SERIES.

In the third question, the search value COMPUTERS is observed to be incorrect and the user deletes the question and enters correctly on the line below.

The $DELETE command for a given question must be entered prior to the beginning of the next question and the PRINT, SORT, REPORT, or HEAD statements for the given question.

5.7 Query Optimization

The processing time for a user query can be optimized by entering the query such that the qualifying constraints are entered in the order of most restricting to least restricting. Consider a personnel data base where the user desires to locate all persons whose job title is computer programmer and salary is greater than $10,000. Assume the user knows that over 40 percent of the persons in the data base make over $10,000 and that about 2 percent of the persons have job titles of computer programmer. The query should be entered with the job title constraint first and the salary constraint last.

Example:

```
IF JOB-TITLE IS COMPUTER PROGRAMMER
AND SALARY GR THAN 10000;
```

In many cases, of course, the user will not know which constraints are most or least restricting; or the information may be known about some but not all of the constraints.

Example:

```
(Most)          IF NAME1 IS A OR B;
(Don't Know)    AND NAME2 IS C;
(Don't Know)    AND NAME3 IS D OR E;
(Least)         AND NAME4 IS F OR G;
```

In this example, the user knows that the constraint "NAME1 IS A OR B" will be most restrictive and the constraint "NAME4 IS F OR G" will be least restrictive. Thus, they are entered first and last, respectively. It is not known which of the other two constraints -- "NAME2 IS C" or "NAME3 IS D OR E" is the most restrictive with respect to each other. They are thus entered in the middle two positions in any arbitrary order.

Note that the conjunction "AND" separates the constraints. No query optimization is performed on constraints separated by "OR".

Example:

```
IF NAME1 IS A
OR NAME2 IS B;
```

In this case, it does not matter whether "NAME1 IS A" or "NAME2 IS B" is the most restrictive constraint. They are entered in either order.

5.8 Queries on Coded Elements

In queries on coded data elements, the assigned code is specified as the search value. On output, the code is translated to the table value unless the user specifies in the query that the coded value is to be output. This is done by use of the keyword "CODED" in print and report statements. In the following examples, assume that N1, N2, and N3 are coded elements and are output in Report A.

```
IF N1 IS 21 OR 24;
PRINT N1, N2, AND N3.
(Table values will be output for all 3 elements)

IF N1 GR 25;
PRINT N1, N2, AND (CODED) N3.
(Only N3 will be output coded)

IF N1 GR 25 OR N2 LT 27;
PRINT (CODED) N1, N2, AND N3.
(All 3 elements will be output coded)

IF CODE-A IS B OR C;
REPORT A (CODED)
(All coded elements in Report A will be output coded)

IF CODE-A IS B;
REPORT A
PRINT (CODED) N1, N2, AND N3.
(Same effect as previous query)
```

```
IF N1 GR 27;
REPORT A
PRINT N1, N2, AND (CODED) N3.
(Report A will output only N3 coded; table values will
 be output for N1 and N2)
```

5.9 Queries on Tree Elements

In queries on tree elements, it is assumed that the user is referencing the tree table value if it is an "equal" type search. This allows the system to perform automatically the hierarchical search in the tree table. If any type search other than "equal" is performed, it is assumed that the user is directly referencing the assigned hierarchical code.

Examples:

```
IF SKILL EQ COMPUTER SOFTWARE;
PRINT NAME AND ADDRESS.
```

In this example, all records in the data base will be selected which contain computer software _or_ any skill under it in the tree table hierarchy (i.e., computer software or all skills which are sub-levels of computer software).

```
IF PREFIX SKILL BE AAABA/AAABB;
PRINT NAME AND ADDRESS
```

In this example, using the skill table shown in Sec. 4.10, a generic search is performed by directly referencing the assigned codes with a ranging (BE) search. All records with computer software, computer hardware, or any of their sub-level entries would be selected. This type of query, where the assigned codes are referenced directly via a non-equal search, is usually not necessary in typical tree queries. The same search could have been generated, for example, with the following query:

```
IF SKILL EQ COMPUTER SOFTWARE
  OR COMPUTER HARDWARE;
```

This query format is preferred, since the user does not have to know the assigned codes and the syntax is more natural.

6. REPORT DEFINITIONS

SHARP has on line and off line report definition capability. Users may define reports on line in an interactive mode. Multiple reports may be defined at one time. If user errors are made in the report definitions, error messages will be output informing the user of the type of error encountered. The user will then be allowed to re-enter the incorrect report definitions correctly.

Each report definition is catalogued under a user specified identification acronym. The user may then select the report at query time by specifying the report ID. A user may define a report and select it in a query during the same terminal session. See Section 5.4 for a complete description of how defined reports are selected during queries.

Users may define both columnar and row type reports. In a columnar report, the information from one data base record appears on a single line on the output report page. Thus, a given data element value would appear in a single column of the report page. In a row type report, multiple lines are required on the output report page to contain the information from one record.

The initial input for all report definitions is the field ID followed by the data base file for which the report is defined (e.g. ID PERSNL). The file ID is followed by the report ID in the form REPORT XXX, where XXX is a one to three character report identification assigned by the user. This sequence will be followed by specific report definition information. If the user desires to define more than one report for the file, the sequence is repeated beginning with REPORT XXX where XXX would be the ID for the next report.

If it is desired to define reports for additional files, the sequence is repeated beginning with a new file ID.

All report definition information is introduced by keywords (ID and REPORT are keywords). Additional keywords are described below.

DBTYPE - Introduces information which describes the record key in the file. The information is inserted as heading information in the report.

MHEAD - Precedes master heading information which will be output at the top of each report page.

PRINT - Precedes the acronyms of the elements to be printed out in the report.

LINE x - Specifies that the elements which follow are to be output on line x.

N - Indicates that the acronym immediately preceding N is to be output with the data value in a row report.

M - Indicates that a textual type data element is to be edited on output. Editing includes preventing the splitting of a word between two lines of output in a row report.

SPACES - Used in a row report to output a line of spaces.

ALL - Introduces a one or two character pattern which is to be output as a line in a row report. (e.g., ALL * would output a line of asterisks. ALL '.-' would output a line of "dot/dashes").

COL HEAD x - Introduces from one to three lines of column heading information which is to be centered over data column x in a column report.

AVERAGE
or
AVG - Precedes the elements, for which the average data value is to be computed and output.

TOTAL
or
SUM - Precedes the element(s) for which the sum of the output data values are to be computed.

PERCENT - Precedes the two elements for which the percentage is to be computed for each output record.

PERCENT SUM
or
PERCENT TOTAL - Precedes the two elements which are to be summed. Then a single percentage is to be computed on the two sums.

MAXIMUM
or
MAX - Precedes the element for which the maximum value is to be computed and output.

MINIMUM
or
MIN - Precedes the element for which the minimum value is to be computed and output

WIDTH = x - Defines the report page width in number of characters (x). (e.g. PAGE WIDTH = 135)

CHANGE or EJECT — Introduces the element whose change in value will trigger a page eject. (e.g. CHANGE PAGE ON FSN)

SIGMA — Precedes the element(s) for which the standard deviation is to be computed and output.

COUNT — Precedes the element(s) whose output data values are to be counted.

SUBTOTAL or SUBTOTALS or SUB-TOTAL — Precedes the element(s) whose output data values are to be subtotaled when a specified data element changes in value.

ON — Immediately precedes the element whose change in value will trigger a computation or a page eject. (e.g. PRINT SUBTOTAL QTY1 ON FSN)

$END — Terminates a report definition sequence.

6.1 Row Report Definition

The following is an example of a typical row type report defined using the keywords described above.

```
ID NAVLIS
REPORT ABS
DBTYPE IS DOCUMENT
WIDTH = 72
MHEAD = 'NAVLIS TECHNICAL DOCUMENTS WITH ABSTRACTS'
LINE 1 = TITLE, M
LINE 2 = ORG/NAME, N, M.
LINE 3 = SECURITY, N, PUBL/DATE, N AND
         LIB/LOC, N.
LINE 4 = ABSTRACT, M.
LINE 5 = ALL -
$END
```

The report is defined for data base file NAVLIS. The report is to be catalogued under the name ABS. "DBTYPE IS DOCUMENT" indicates that the logical record key of the file is to be output with each record and is to be given a heading of DOCUMENT. In this case, the record key is automatically output on the right side of the page annotating each record. If the user does not wish the record key to be output at all, the DBTYPE statement is omitted. If he wishes it output, but not in the default position, he omits the DBTYPE statement and treats it like any other element by naming it in an appropriate LINE or PRINT statement.

The user indicates the width of the output page to be 72 characters. Seventy two characters is the width of a normal teletype page and is considered to be the default. If no WIDTH specification is entered, 72 will be assumed.

The master heading is indicated by the MHEAD specification. A maximum of 60 characters of master heading information may be entered. The information is bounded by either single quotation marks or single dollar signs. The heading will form a single line centered at the top of each output report page.

Following the heading specification, the LINE specifications are entered which indicate what elements are to be output on each line of the report. The elements are indicated via their assigned acronyms. Line 1 is to contain the element TITLE. The M indicates that the title may exceed one report line and it is to be edited such that words are not split between lines and extraneous blanks are omitted. Line 2 is to contain ORG/NAME. The N indicates that the element ORG/NAME is to be output annotated by its acronym. (e.g., ORG/NAME: NAVAL SHIP ENGINEERING CENTER). The M indicates that a text edit is also to be performed on ORG/NAME. On line 3, three elements are to appear from left to right in the stated order. Each element is to be annotated by its acronym. Line 4 is to contain the element ABSTRACT which is to be text edited. Text editing should be performed only on textual elements whose length may exceed the report page width. Line 5 is to contain all dashes. The ALL specification allows a user to specify separators between blocks of information for readability. In this case, the user is separating the information from each data base record with a line of dashes. $END terminates the report definition sequence.

English-like punctuation is allowed. Commas or the conjunction AND may be used to separate elements in a line specification. A period is optional at the end of each line specification.

The following is a typical output hit report for the above report definition.

```
              NAVLIS TECHNICAL DOCUMENTS WITH ABSTRACTS

REPORT ABS JJ001                                    MARCH 20 1974

                                                         DOCUMENT

(U) SECAS VALIDATION REPORT - SUMMARY OF SHIP                3203
VALIDATIONS PERFORMED
ORG/NAME:  NAVAL SHIP ENGINEERING CENTER
SECURITY:  U PUBL/DATE: MAR 15 73  LIB/LOC: 1806
A LIST OF SHIPS THAT ARE VALIDATED UNDER
SECAS PROGRAM PROCEDURES
------------------------------------------------------

(U) SHIPS EQUIPMENT CONFIGURATION ACCOUNTING                 3308
SYSTEM (SECAS-ADP OPERATIONS MANUAL 500K 1,
VOL. 3)
SECURITY:  U PUBL/DATE: MAY 11 73  LIB/LOC: 1806
A FLEET SUPPORT DOCUMENT CONTAINING THE
COMPLETE SYSTEM/EQUIPMENT IDENTITY ON NON
EXPENDABLE ORDNANCE AND LOGISTICS SUPPORT
------------------------------------------------------
```

In the above sample report, two hit records with record keys 3203 and 3308 are output. Note that the line numbers which some of the elements begin on do not match the line numbers specified in report definition when multiple lines are required to contain a single element. In the first record, two lines are required for TITLE. Thus, ORG/NAME begins on line 3. The user need not be concerned with the actual number of lines which may appear for a data element. The system automatically adjusts the line numbers on which elements are output when variable length information is output which may exceed one line.

In the second output record, no ORG/NAME was output because there was no data value entered for ORG/NAME in that data base record. The record keys are output on the right under the heading specified by the DBTYPE statement. The report ID appears in the subheading. JJ is the query ID of the user who submitted the query in which the report was selected. 001 is the question number in the query. The date the report is generated appears on the right of the subheading. If variable heading information had been input with the query it would have appeared directly under the master heading. Note that TITLE and ABSTRACT are not annotated by their acronyms while the other elements are.

The system will determine the character position of multiple elements on a line by adding the following information for each element on the line.

a) The number of characters in the acronym if annotation is requested (N specification).

b) Two characters between the acronym and data value if N specification is used.

c) The maximum number of characters a data value may have as specified in data base definition.

d) Two spaces between one data value and the beginning of the next.

If a user defines multiple data elements to be output on a line, and the amount of data is too large to be contained on the line, the data will overflow to the following line on the output report. If the total number of characters exceeds the page width, then multiple lines will be required to contain the information.

A maximum of 60 lines may be defined in a row report containing a maximum of 100 data elements.

Whenever a row report is desired which would contain one data element per line, it is not necessary to define each line separately. The lines may be defined with a print statement.

Example:

```
ID NAVLIS
REPORT AUT
DBTYPE IS DOCUMENT
MHEAD = 'NAVLIS TECHNICAL DOCUMENTS'
PRINT  TITLE, M, ORG/NAME, M, AND
       ABSTRACT, N, M.
$END
```

In this case, TITLE would be output first beginning on the first line. Following TITLE, ORG/NAME would be output beginning a new line. Then abstract would be output following ORG/NAME and beginning on a new line.

6.1.1 Repeating Elements in Row Reports

Users may specify the location of data values for repeating elements by repetition of the acronym in the appropriate line or print statement.

Example:

```
ID NAVLIS
REPORT  A2  DBTYPE IS DOCUMENT
MHEAD = 'NAVLIS AUTHOR REPORT'
LINE 1 = AUTHOR AND AUTHOR.
LINE 2 = AUTHOR AND PUBL/DATE.
LINE 3 = TITLE, M.
REPORT  A3  DBTYPE IS DOCUMENT
MHEAD = 'NAVLIS AUTHOR REPORT 2'
PRINT  AUTHOR, AUTHOR, AUTHOR,
       PUBL/DATE AND TITLE, M.
$END
```

In this example, two reports are defined for the same data base file. AUTHOR is a repeating element which may have up to three entries in a data base record. For report A2, the first two authors entered are to be output on line 1. The third author will be output on line 2 along with the publication date. For report A3, the three authors will be output on separate lines.

6.1.2 Computations In Row Reports

Computations may be specified in the definitions of both row and column reports. In row reports, all computations except PERCENT are specified in a PRINT statement. The PERCENT may be specified in either a LINE or PRINT statement. See Section 5.3.4 for a complete description of how computations are specified in PRINT statements.

When computations are specified in report definitions, the computations will always be output whenever the report is selected. Computations may be output in a report in which they were not defined by specifying the computations at query time (see Section 5.4).

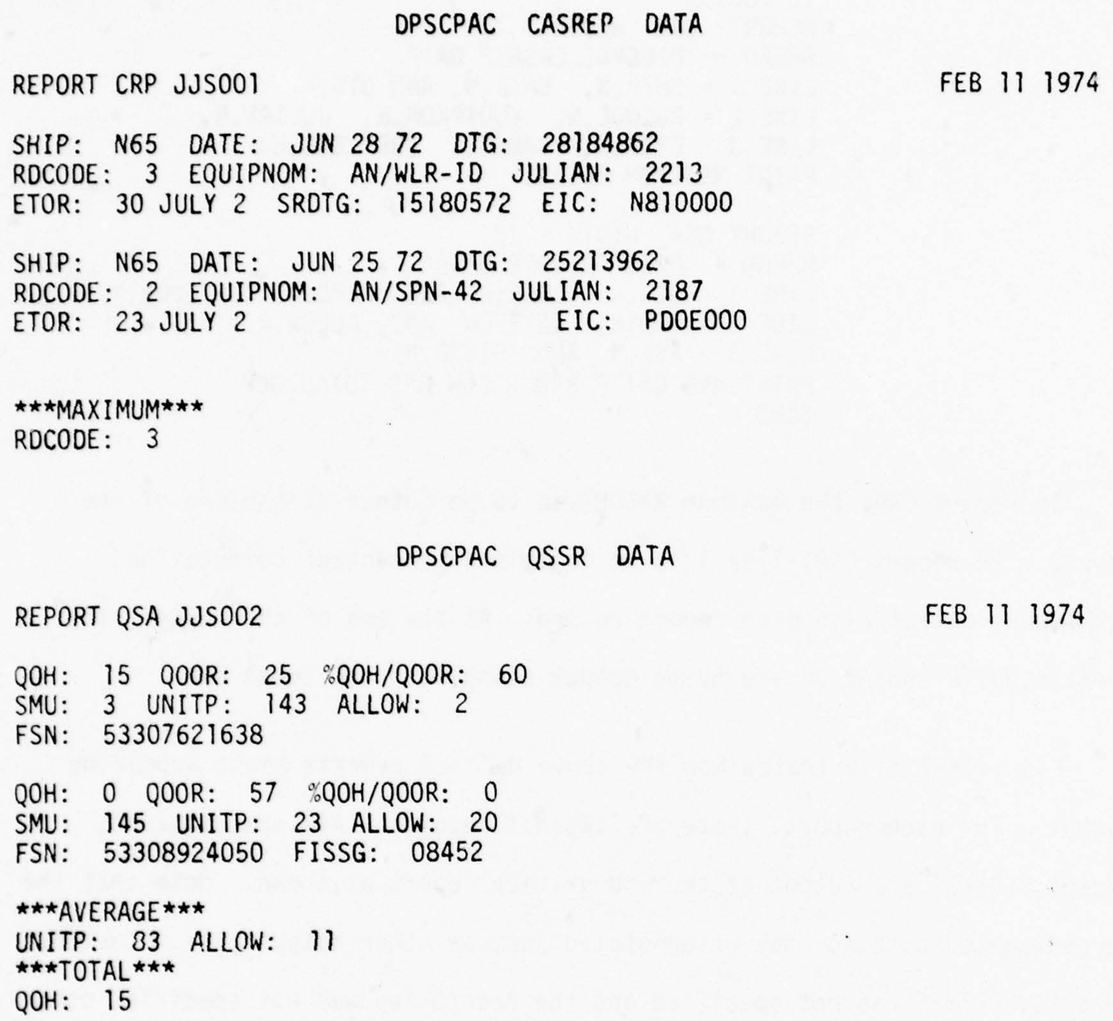
The following examples illustrate computations defined for row reports.

```
ID LOGIS2
REPORT  CRP  WIDTH = 72
MHEAD = 'DPSCPAC CASREP DATA'
LINE 1 = SHIP,N,  DATE,N, AND DTG,N.
LINE 2 = RDCODE,N,  EQUIPNOM,N,  JULIAN,N.
LINE 3 = ETOR,N,  SRDTG,N  AND  EIC,N.
PRINT MAXIMUM RDCODE

REPORT QSA  WIDTH = 72
MHEAD = 'DPSCPAC QSSR DATA'
LINE 1 = QOH,N,  QOOR,N  AND  PERCENT QOH/QOOR,N.
LINE 2 = SMU,N,  UNITP,N  AND  ALLOW,N.
LINE 3 = FSN,N  AND  FISSG,N
PRINT AVG UNITP AND ALLOW AND TOTAL QOH
$END
```

In report CRP, the maximum RDCODE is to be output at the end of the report. In report QSA, line 1 is to contain a percentage computation (QOH/QOOR) output with each report record. At the end of the report, the average UNITP and ALLOW are to be output along with the total QOH.

Figure 6-1 illustrates how the above defined reports might appear on output. For each report, there are two hit records. All computations, except PERCENT are output at the end of each report as shown. Note that the percentage computation may be annotated just as other outputs are. In these reports, DBTYPE was not specified and the record key was not specified otherwise in the report definitions, so it is not output.

Computations in both row and column reports may be specified on elements not defined for output in the body of the report. For example, a user could specify the average value of a field be computed and output without specifying that the values which were used in the computation be output.



```
                        DPSCPAC  CASREP  DATA

REPORT CRP JJS001                                           FEB 11 1974

SHIP:  N65  DATE:  JUN 28 72  DTG:  28184862
RDCODE:  3  EQUIPNOM:  AN/WLR-ID  JULIAN:  2213
ETOR:  30 JULY 2  SRDTG:  15180572  EIC:  N810000

SHIP:  N65  DATE:  JUN 25 72  DTG:  25213962
RDCODE:  2  EQUIPNOM:  AN/SPN-42  JULIAN:  2187
ETOR:  23 JULY 2                    EIC:  PDOE000


***MAXIMUM***
RDCODE:  3



                        DPSCPAC  QSSR  DATA

REPORT QSA JJS002                                          FEB 11 1974

QOH:  15  QOOR:  25  %QOH/QOOR:  60
SMU:  3  UNITP:  143  ALLOW:  2
FSN:  53307621638

QOH:  0  QOOR:  57  %QOH/QOOR:  0
SMU:  145  UNITP:  23  ALLOW:  20
FSN:  53308924050  FISSG:  08452

***AVERAGE***
UNITP:  83  ALLOW:  11
***TOTAL***
QOH:  15
```

Figure 6-1 - Sample Row Reports With Computations

Example:

```
ID  BASE2
REPORT A21
MHEAD = 'SAMPLE REPORT'
LINE 1 = CODE-A,N,  QTY1,N AND QTY2,N.
LINE 2 = CODE-B,N,  CODE-C,N,  AND  CODE-D,N.
PRINT AVERAGE QTY3.
$END
```

6.2 Column Report Definition

Column reports are normally defined when the information to be selected from a data base record can be contained on a single line of the output page. In column reports, no LINE statements are used. Also, the M, N, and ALL specifications are not used. Column reports are identified by the keywords COL HEAD or COLHEAD which introduce column headings for the report. The following is an example of a typical user defined column report.

```
ID SHIPOH
REPORT A  DBTYPE IS HULL-ID
MHEAD = 'SHIP OVERHAUL AND REPAIR REPORT-WITH
         FUNDING STATUS'
COL HEAD 1 = 'START/DATE'
COL HEAD 2 = 'FCLTY'
COL HEAD 3 = 'SHIP/ID'
COL HEAD 4 = 'FNDG/PRIOR/YEAR'
COL HEAD 5 = 'CRNT/YEAR/PLAN'
PRINT STARTDATE, LOC, SHIPID, PYRYRFND,
         AND CNTYRPLN
PRINT SUM PYRYRFND AND CNTYRPLN
PRINT MAX PYRYRFND AND CNTYRPLN
         AND MIN PYRYRFND AND CNTYRPLN.
$END
```

In this case the user desires the record key, which is HULL-ID, to be output automatically on the right hand side of the page with HULL-ID as the column heading.

Following the master heading are the column headings for the respective columns of data. The left most column heading for the report page is entered first (COL HEAD 1). Each column heading is bounded by either single quotation marks or single dollar signs. Slashes are used to separate column heading lines. A maximum of three column heading lines per column heading may be entered with a maximum of twelve characters per column heading line. The first column heading will consist of two heading lines with the START on the first line and DATE on the second. Following the column heading, a Print statement is entered to list the data elements which are to be output under the column headings. The element names are entered via their acronyms, the first element to be output under column heading one, the second under column heading two, etc.

Following the print statement which defines the data columns, the desired computations are specified via a print statement. In this case, the elements to be output in the fourth and fifth columns (PYRYRFND and CNTYRPLN) are to be summed. The maximum and minimum values are to be computed for the same two elements.

The following is a typical output report for the above report definition showing three hit records.

```
          SHIP OVERHAUL AND REPAIR REPORT-WITH FUNDING STATUS

REPORT  A  LDM001                                             APRIL 4 1972

START          FCLTY          SHIP          FNDG           CRNT          HULL-ID
DATE                           ID           PRIOR          YEAR
                                            YEAR           PLAN

APR 75          C          DEWEY            205            7205          DLG 14
APR 75          5          PUGET SOUND       15            3335          AD  38
MAY 75          5          WELCH              0             679          PG  93

                              ***TOTAL***
                                            220           11219

                              ***MAXIMUM***
                                            205            7205

                              ***MINIMUM***
                                              0             679
```

Each data column in the report is centered with respect to its column heading. The columns are spaced evenly across the page for the page width defined (in this case, 72 is default). Note that since HULL-ID is specified for automatic inclusion in the output report by the DBTYPE specification, it is not otherwise shown in the report definition. It is automatically included as the sixth output column.

The data values for the two computational elements (PYRYRFND and CNTYRPLN) are automatically zero suppressed and right adjusted in their fields. The other information is left adjusted with respect to the field width.

The column headlines are top adjusted. They could have been bottom adjusted through use of the slashes as shown below.

```
COL HEAD 1 = '/START/DATE'
COL HEAD 2 = '//FCLTY'
COL HEAD 3 = '/SHIP/ID'
COL HEAD 4 = 'FNDG/PRIOR/YEAR'
COL HEAD 5 = 'CRNT/YEAR/PLAN'
```

The above column headings would be aligned as follows.

```
                                            FNDG       CRNT
START                       SHIP            PRIOR      YEAR
DATE          FCLTY          ID             YEAR       PLAN
```

The slashes allow the user to align column heading lines as desired.

Computations which are defined for column reports will always appear in a PRINT statement. The computation PERCENT is the only computation which may be defined as one of the data columns.

Example:

```
COL HEAD 1 = 'QTY/1'
COL HEAD 2 = 'QTY/2'
COL HEAD 3 = 'PERCENT/QTY 1 OF/QTY2
PRINT QTY1, QTY2, AND PERCENT QTY1/QTY2
PRINT MAXIMUM QTY2
```

If a column report is defined which is too wide for the defined page width an error diagnostic will be output. In the interactive mode, the user may redefine the report correctly. The following technique should be used to compute the minimum width required for the report.

a) For each data column, take the number of characters in the widest column heading line <u>or</u> the maximum number of characters which may be output for a data value, whichever is larger.

b) Sum the numbers computed for each data column.

c) Add one space for each column except the last (minimum spacing between columns)

c) Compare the grand total to the page width. If larger than the page width, the report definition is invalid.

6.3 Miscellaneous Constraints

The ALL specification, used in row reports for outputting a one or two character pattern on a line, requires that a two character pattern be bounded by single quotation marks.

Example:

```
LINE  6 = ALL *

LINE  10 = ALL '.*'
```

Line 6 would consist of all asterisks, while line 10 would consist of the two character pattern "period/asterisk" across the line.

In row reports, a line of spaces may be output at any line position desired.

Example:

```
LINE 3 = DATE AND LIB/LOC
LINE 4 = ALL SPACES
LINE 5 = ABSTRACT, M
LINE 6 = ALL SPACES
```

In column reports, if computations are specified on data elements not appearing in the body of the report, the computed values will appear at the end, annotated by their acronyms in a manner similar to the row report example shown in Figure 6-1. Otherwise, the computations will be output directly under the column values in column reports.

The key work ON may be used to specify that a computation or page eject occurs only when an element changes in value.

Example:

```
CHANGE PAGE ON FSN
PRINT MAX QTY1 AND QTY2 ON FSN
PRINT AVG QTY3 ON FSN
PRINT SUBTOTAL QTY4 ON FSN
```

In this example, FSN is the variable whose change will cause the specified computations to occur. ON must be used with the SUBTOTAL computation. It is optional with other computations. Note that the output should normally be sorted on the change variable if the computation is to be meaningful. Only one change variable may be entered following the keyword ON.

The following computations can be performed only on real or integer data elements as defined in data base definition- TOTAL, AVERAGE, PERCENT, PERCENT SUM, SUBTOTAL, and SIGMA. The computations MAXIMUM, MINIMUM, and COUNT can be performed on data elements of all formats.

The user may specify that computations or page change occur when a prefix or partial part of an element changes in value.

Example:

```
CHANGE PAGE ON (3)FSN
PRINT SUBTOTAL QTY1 ON (2)CODE1
PRINT AVG QTY2 ON (*2,3)CODE2
```

In this example, the report page is to be ejected when the first 3 characters of FSN change in value. QTY1 is to be subtotaled when the first 2 characters of CODE1 change in value. QTY2 is to be averaged when characters 3 through 5 of CODE2 change in value.

Example:

```
ID PERSNL
REPORT P10
PAGE WIDTH = 72
MHEAD = 'PERSONNEL LOCATION REPORT'
COL HEAD 1 = 'JOB/CODE'
COL HEAD 2 = 'JOB/SERIES'
COL HEAD 3 = 'GRADE'
COL HEAD 4 = 'SALARY'
PRINT CODE, SERIES, GRADE, AND SALARY.
CHANGE PAGE ON (3)CODE
PRINT AVG GRADE AND SALARY
    ON (3)CODE
$END
```

In this example, Job Code could be a hierarchical code, where the first 3 digits represent the division number. The report would output the average grade and salary in each division. Information for each division would begin on a new page.

For Report definitions containing coded or tree data elements, the maximum size of the table value entries should be assumed for determining the space required on the output reports. Then, either the assigned code or the table value will "fit" the allocated space. The option to output either assigned code or table value is available only with coded elements, not tree elements. For tree elements, the table values are always output in reports.

7. PROCEDURAL LANGUAGE INTERFACE

7.1 General Characteristics

THE SHARP data base consists of variable length logical records. Each logical record may contain a variable number of data elements. By defining the data elements as logical parts of the data record, each application program can specifically request only the data elements pertinent to that program. The process of element selection is accomplished by defining an acronym list in the application program. The acronym list is made up of literal constants representing the element names as they were defined in the Data Definition. The element names within this list may be in any sequence and there may be a variable number of acronym lists within each application program. Because the program requests only the elements needed for processing, when changes to non-pertinent data elements or additions of new data elements occur, the program is not affected; therefore, eliminating reprogramming, recompiling and relinking. The procedural language interface contains all the coding necessary to access the SHARP data files. Therefore, the application program is not required to define SHARP files or execute actual I/O functions.

SHARP logical records may be requested serially, sequentially, or randomly from any SHARP file. The file to be accessed and the type of retrieval is dependent on the setting of the file-name and a one-position function code in the application program. If the records are to be retrieved randomly, the record key must be defined along with the file name and function.

7.2 Linkage

The SHARP procedural language interface can communicate with any programming language that supports standard linkage procedures. This communication is accomplished by defining a parameter list in the application program. The procedural language interface then analyzes this parameter list to determine the action requested.

Following is an example of linkage, in COBOL form, and a discussion of each function in the parameter list.

```
    CALL 'SHARP PLI' using operation, file-name, element-list,
                     record-area, error-eof-ind, password, record-key.
    ENTER COBOL.
```

There can be multiple call statements within an application program. The parameters in the call statement must be in the same sequence as shown above.

The following parameters would be defined in the WORKING-STORAGE SECTION.

Operation - is a one-character code which indicates the access method to be used on the SHARP file. The valid codes and their function are:

1 - read SHARP file serially
2 - read SHARP file randomly
3 - read SHARP file sequentially

A serial request causes two different results; one, if the record key is nine characters or less, the records will be read in record key sequence; two, if the record key is greater than nine characters, the records will be read in accordance to their physical location on the file (i.e., in hash key sequence). A random request causes a specific record to be read dependent on the contents of record key in the application program. A sequential request causes the records to be read in record key sequence.

An application program may request a combination of operations against the same file. For example, a random request can be executed to get to a certain point in the file, then records can be requested sequentially from that point.

File-Name - the six-character file name as defined in the SHARP data definition.

Element-List - this parameter references a list which contains the acronyms assigned to the elements during the SHARP data base definition. The acronyms in the list can be in any sequence and there can be any number of element lists within one application program. Each element-list is terminated by a "$END".

Different elements can be requested from the same file at any time by executing other call statements which reference the different element lists. It is up to the application program to ensure that when referencing another element list the proper record-area is specified. All names in every element list must be from the file defined in the file-name parameter.

Record-Area - is the name of the user area into which the requested data elements are to be read. The elements will be placed in the user area in the order specified by the element list. The user is responsible for defining the record area so that it will accommodate the sequence of the element-list and the data requirements.

The length of the record area can be calculated by totaling the lengths of each element in the element list. If repeating, link or link-role, elements are requested the user must ensure that the structure of the record area will accommodate the maximum number of occurrences.

<u>Error-Eof-Ind</u> - is the name of a one-character encoded field which is returned to the application program indicating the result of the request. The SHARP Procedural Language Interface always returns control to the calling program with one of the following condition codes:

Ø - indicates a good request.

1 - indicates an error occurred. It is the responsibility of the application program to determine the action to be taken at this point.

3 - indicates an end of file condition.

<u>Password</u> - is a ten-character field which indicates that the user has been authorized to access the requested SHARP file.

<u>Record-Key</u> - is the name of an area which contains the actual key value of the requested record. This field must be specified when the operation code indicates a random request. The record key is originally defined in the SHARP data definition as element number zero.

7.3 Procedure Language Interface Examples

The file being used in this demonstration is the Personnel File. The File-ID is PERS. The Record-Key is OPRNO. Although all the examples are COBOL oriented, the same fields must be defined no matter what language the calling program is written in.

The following elements are in the SHARP Personnel record:

DATA NAME	ACRONYM	CHARACTERS
Operator Number	OPRNO	3
Name	NAME	20
Address	ADDRESS	20
City	CITY	11
Zip Code	ZIPCODE	5
Social Security Number	SSN	9
Spouse's First Name	SPOUSE	9
Previous Positions Held	PREVPOS	10
Previous School Courses	COURSES	30

7.3.1 Sequential Request Example

```
WORKING-STORAGE SECTION

01  OPERATION           PIC 9.
01  FILE-NAME           PIC X(10) VALUE "PERS".
01  ELEMENT-LIST
    05  FILLER          PIC X(30) VALUE "OPRNO".
    05  FILLER          PIC X(30) VALUE "NAME".
    05  FILLER          PIC X(30) VALUE "ADDRESS".
    05  FILLER          PIC X(30) VALUE "CITY".
    05  FILLER          PIC X(30) VALUE "ZIPCODE".
    05  FILLER          PIC X(30) VALUE "$END".
01  RECORD-AREA         PIC X(59).
01  ERROR-EOF-IND       PIC X.

PROCEDURE DIVISION.

GET-SEQUENTIAL.

    MOVE 3 TO OPERATION.

    ENTER LINKAGE.

    CALL SHARPPLI USING OPERATION, FILE-NAME, ELEMENT-LIST,
                        RECORD-AREA, ERROR-EOF-IND.

    ENTER COBOL.
```

NOTE. Both Password and Record-Key are omitted from the parameter list because for a sequential or serial request they are optional.

If the request was successful, Error-Eof-Ind would contain a zero and Record-Area would contain the following:

```
[000/Doe John     /1000 State Street    /San Diego    /92071]
 1  4              24                    44             55
```

NOTE. This same example could be used for a serial request by moving a one to Operation instead of a three.

7.3.2 Random Request Example

```
WORKING-STORAGE SECTION.

01  OPERATION               PIC 9.
01  FILE-NAME               PIC X(10) VALUE "PERS".
01  ELEMENT-LIST
    05  FILLER              PIC X(30) VALUE "SSN".
    05  FILLER              PIC X(30) VALUE "NAME".
    05  FILLER              PIC X(30) VALUE "SPOUSE".
    05  FILLER              PIC X(30) VALUE "$END".
01  RECORD-AREA             PIC X(38).
01  ERROR-EOF-IND           PIC X.
01  PASSWORD                PIC X(10) VALUE SPACES.
01  RECORD-KEY              PIC X(3).
PROCEDURE DIVISION.
GET-RANDOM.
    MOVE 2 TO OPERATION.
    READ KEY-FILE AT END . . . .
    MOVE IN-KEY TO RECORD-KEY.
    ENTER LINKAGE.
    CALL SHARPPLI USING OPERATION, FILE-NAME, ELEMENT-LIST,
                        RECORD-AREA, ERROR-EOF-IND, PASSWORD,
                        RECORD-KEY.
ENTER COBOL.
```

NOTE. Password must be specified because the entries in the parameter list are positional and Record-Key is mandatory for a random request.

If the request was successful Error-Eof-Ind would contain a zero and Record-Area would contain the following:

```
[111223333/Doe John           /Jane     ]
 1        10                  30
```

In the previous examples it is assumed that all elements requested are present in the record. If an element is not present that field would contain the appropriate fill characters such as spaces or zeroes.

7.3.3 Combination Random/Sequential Request Example

In this example all individuals with an operator number of 200 or greater would be retrieved.

```
WORKING-STORAGE SECTION.

01  OPERATION              PIC 9.
01  FILE-NAME              PIC X(10) VALUE "PERS".
01  ELEMENT-LIST
    05  FILLER             PIC X(30) VALUE "OPRNO".
    05  FILLER             PIC X(30) VALUE "NAME".
    05  FILLER             PIC X(30) VALUE "$END".
01  RECORD-AREA            PIC X(23).
01  ERROR-EOF-IND          PIC X.
01  PASSWORD               PIC X(10) VALUE SPACES.
01  RECORD-KEY             PIC X(3).
```

```
PROCEDURE DIVISION.

GET-RANDOM.
    MOVE 2 TO OPERATION.
    MOVE "200" TO RECORD-KEY.
    PERFORM CALL-SHARPPLI THRU CALL-EXIT.
    MOVE 1 TO OPERATION.
GET-SEQUENTIAL.
    PERFORM CALL-SHARPPLI THRU CALL-EXIT.
                .
                .
                .
                .
    GO TO GET-SEQUENTIAL.
CALL-SHARPPLI.
    ENTER LINKAGE.
    CALL SHARPPLI USING OPERATION, FILE-NAME, ELEMENT-LIST,
                        RECORD-AREA, ERROR-EOF-IND, PASSWORD,
                        RECORD-KEY.
    ENTER COBOL.
```

NOTE. The same call statement can be used for a combination of requests by changing the value of Operation.

If the request was successful, Error-Eof-Ind would contain a zero and Record-Area would contain the following:

```
[200/Smith John        ]
```

If the request was not successful Error-Eof-Ind would contain a one and no data would have been moved to Record-Area. At this point it is up to the application program to decide what action is to be taken.

7.3.4 Example of Retrieving a Field Defined as Repeating

The requested field is Previous Positions Held. There are a maximum of ten entries and each entry is ten positions in length. The record-area must be defined to accommodate all ten entries.

If there were three entries present in the SHARP record the area reserved in Record-Area would contain the following:

```
[OPERATOR  /PROGRAMMER/ANALYST   /                              ]
```

The remainder of the area reserved for the other seven entries would be spaced filled.

7.3.5 Example of Retrieving a Field Defined as Link, Role

The field is Previous School Courses. This field is broken down by months and can contain up to three courses completed per month. The course field is 30 positions in length. The Record-Area must be defined to accommodate the maximum number of entries (12) with the maximum number of occurrences (3). In this case the defined area would be 1080 characters in length (30 x 3 x 12).

If the individual completed two courses in the first month, none in the second, and one in the third, the area reserved in Record-Area would contain the following:

```
[English          /Elementary Algebra      /                              ]
[                 /                        /                              ]
[Cobol            /                        /                              ]
```

All unused occurrences and entries would be spaced filled.

8. SYSTEM MONITOR/USER INTERFACES

SHARP is currently operational on the CDC 6700 Computer System at NSRDC and the UNIVAC Series 70 Computer Systems at DPSCPAC and DPSCLANT. The user/monitor interfaces are somewhat different on the two computer systems. Also, the batch terminal capability described in Section 5, which is available on the CDC System, is not currently available on the UNIVAC Series 70.

8.1 Spectra 70 Monitor/User Interface

General Characteristics

The monitor on the Spectra 70 is designed to perform message handling and loading of requested SHARP programs.

There are three types of messages that are handled by the monitor. The first is the dialog which is a means of communicating between the monitor and the user at the terminal. The second is the input to SHARP which the monitor places on a disk file for later use by the SHARP system. And the third is the output from SHARP which the monitor reads from a SHARP output file and relays to the terminal.

The user indicates to the monitor, via the dialog, which subsystem is to be loaded. Refer to COMMAND and OPTION in the dialog example.

Dialog Example

All dialog input to the monitor must be in the exact format as shown in the following example. If an inquiry received by the monitor is in error, a response is returned to the terminal requesting that the previous inquiry be re-sent. Monitor will execute the above procedure three times before requesting that the user logout.

In this on line example the user is DPSCPAC, the subsystem is SHARP IQUERY, and the query is on the ENGINES data base. Output from the monitor is underlined. Other information is entered by the user.

DIALOG	REMARKS
LOGIN, DPSCPAC	Must be first message. Monitor checks to see if the user has been authorized access to the system.
COMMAND-SHARP	Indicates that SHARP is the requested subsystem. Monitor also handles other systems which are not related to SHARP.
SHARP SUBSYSTEM ENTERED	Indicates which subsystem was selected.
????OPTION-IQUERY	Indicates that the user wants to run an Interactive Query. The results of the request will be printed on line. Other options are BQUERY, REPDEF, and MAINT.
YOU HAVE SELECTED THE INTERACTIVE QUERY OPTION	Indicates which option was selected.
WOULD YOU LIKE AN EXPLANATION OF THE SYSTEM? - NO	

DIALOG	REMARKS
PLEASE SELECT ONE OF THE FOLLOWING DATA BASES	In this step, the monitor checks to see if the user has been authorized to access the selected data base.
AMO MASTER	
ENGINES MASTER	
??ENGINES MASTER	
PLEASE START ENTERING QUERIES ONE STATEMENT	The maximum length of a statement is 71 characters.
AT A TIME. ENTER END TO INDICATE LAST	
STATEMENT.	
IF SMU GR THAN 100?	
PRINT UIC, FSN, SMU.	
$END	
DO YOU WANT YOUR INPUTS PROCESSED?	At this point the user should make sure that the statements entered were valid. If there were any errors in the query, respond no. Monitor will then ignore the last query and allow the user to re-enter it correctly. If there were no errors, reply yes and monitor will load SHARP to process the queries.
PLEASE REPLY YES OR NO	
YES	

After SHARP has processed the queries, the monitor will print the report on the terminal, 12 lines at a time. After every twelfth line monitor will display - "would you like to see more, answer yes or no". If you answer yes monitor will print 12 more lines. If you answer no, monitor will respond with the following.

```
DO YOU HAVE MORE QUERIES?

PLEASE ANSWER YES OR NO

NO

PLEASE LOGOUT

LOGOUT

LOGOUT (TIME, DATE)
```

If you answer yes to this statement, monitor will come back with COMMAND-. If you answer no, monitor will request that you logout.

The above dialog illustrates the IQUERY (interactive query option). Other on line commands available to the SHARP user on the SPECTRA 70 are UPDATE which allows on line file maintenance and REPDEF which allows on line report definition. The user/monitor interfaces for these commands are essentially the same on the SPECTRA as on the CDC 6700. See Section 8.2 for examples of user/monitor dialog during execution of these commands.

8.2 CDC 6700 Monitor/User Interface

SHARP operates on the CDC 6700 under the COMRADE monitor. COMRADE controls the execution of the SHARP DMS. The following commands are available to on line SHARP users on the CDC 6700:

```
IQUERY - Interactive Query
QUERY  - Batch Terminal Query
UPDATE - On line File Maintenance
REPDEF - On line Report Definition
```

To illustrate the usage of the above commands, a typical data base is shown through the sequential stages of data base management - data base definition, file maintenance, report definition, interrogation, and report generation.

Figure 8-1 illustrates the assignment of acronyms for a Ship Overhaul Data Base which contains ship overhaul and funding status. Figure 8-2 illustrates the data base definition. This function is executed off line via inputting the data definition parameters on punched cards.

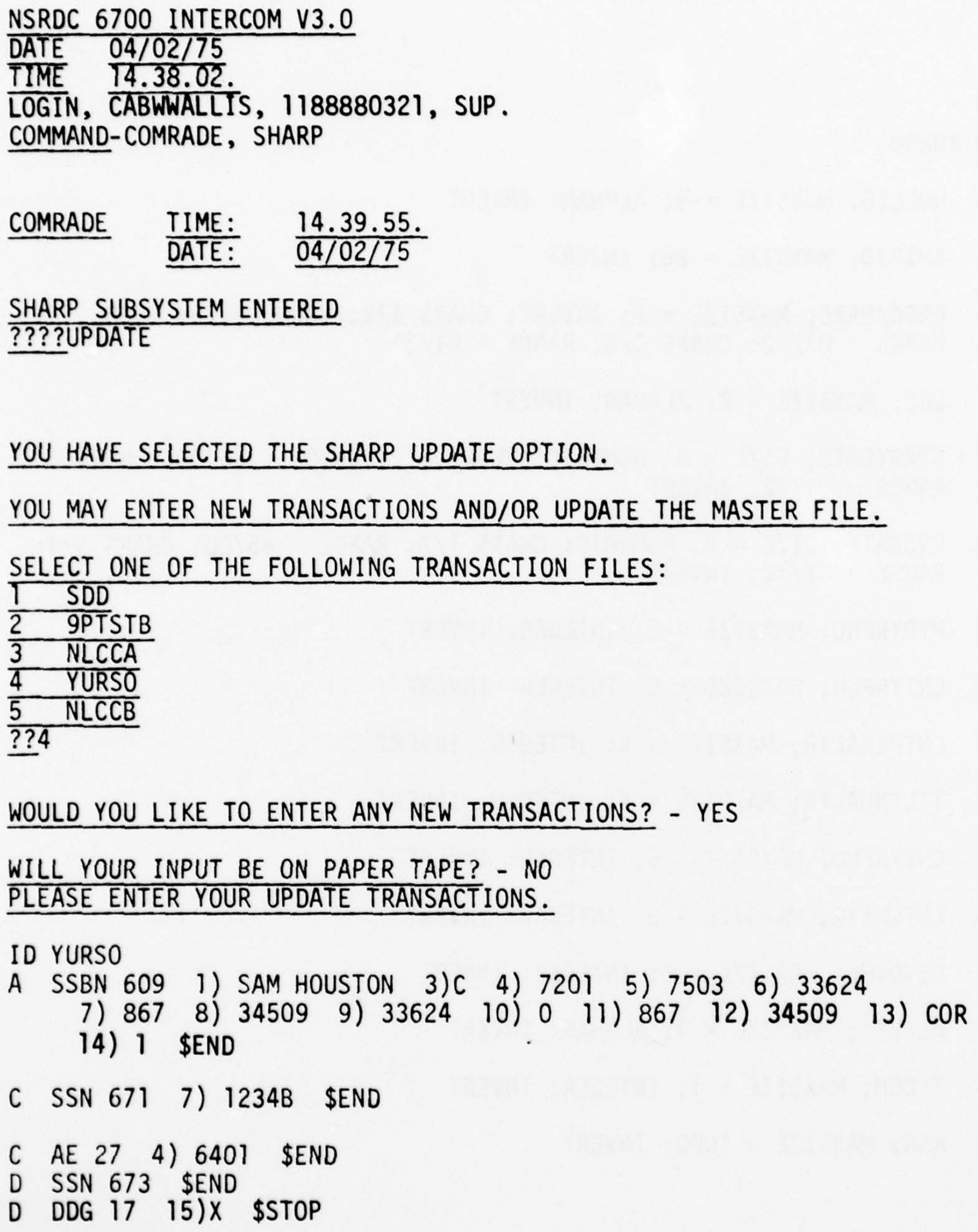
Figure 8-3 illustrates the usage of the interactive command UPDATE. After the log in procedure, the user brings the SHARP DMS into execution by entering COMRADE, SHARP following the computer output COMMAND. The user then types in the command UPDATE following the computer output of four question marks. At this point, the system checks an access table to determine what files the user is allowed to update. He is allowed to select one of the files following the outputting of two question marks. Transaction file four is selected which is the transaction file for the data base

DATA ELEMENTS	ACRONYM
HULL IDENTIFICATION	HULLID
SHIP NAME	SHIPID
PROCESSING DATE	PROC/DATE
OVERHAUL FACILITY	LOC
START OVERHAUL DATE	STARTDATE
END OVERHAUL DATE	ENDDATE
PRIOR YEAR FUNDING	PYRYRFND
CURRENT YEAR PLANNED FUNDING	CNTYRPLN
CURRENT PLAN ALL YEARS	CNTPLNALYR
TOTAL FUNDING ALL YEARS TO DATE	TTLFNDALYR
CURRENT YEAR FUNDING TO DATE	CNTYRFND
ESTIMATED CURRENT YEAR REQUIREMENT	ESTCNTYR
ESTIMATED THROUGH OVERHAUL	ESTOVHL
ACTIVITY CARRY-OVER REQUIREMENTS	ACTVTY
TYPE COMMANDER	TYCOM
NARRATIVE COMMUNICATION MESSAGE	MSG

Figure 8-1 - Data Base Example Ship Overhaul Data Bank

```
ID YURSO

 0)  HULLID; MAXSIZE = 9; ALPHAN; INVERT

 1)  SHIPID; MAXSIZE = 26; INVERT

 2)  PROC/DATE; MAXSIZE = 6; INVERT; CHARS 1/2; VALUE = 75; CHARS 3/4;
     RANGE = 01/12; CHARS 5/6; RANGE = 01/31

 3)  LOC; MAXSIZE = 2; ALPHAN; INVERT

 4)  STARTDATE; SIZE = 4; NUMERIC; CHARS 1/2; RANGE = 65/80; CHARS 3/4;
     RANGE = 01/12; INVERT

 5)  ENDDATE; SIZE = 4; NUMERIC; CHARS 1/2; RANGE = 65/80; CHARS 3/4;
     RANGE = 01/12; INVERT

 6)  PYRYRFND; MAXSIZE = 5; INTEGER; INVERT

 7)  CNTYRPLN; MAXSIZE = 5; INTEGER; INVERT

 8)  CNTPLNALYR; MAXSIZE = 5; INTEGER; INVERT

 9)  TTLFNDALYR; MAXSIZE = 5; INTEGER; INVERT

10)  CNTYRFND; MAXSIZE = 5; INTEGER; INVERT

11)  ESTCNTYR; MAXSIZE = 5; INTEGER; INVERT

12)  ESTOVHL; MAXSIZE = 5; INTEGER; INVERT

13)  ACTVTY; MAXSIZE = 4; ALPHAN; INVERT

14)  TYCOM; MAXSIZE = 1; INTEGER; INVERT

15)  MSG; MAXSIZE = 1000; INVERT
```

Figure 8-2 - Data Base Definition Ship Overhaul Data Bank



```
NSRDC 6700 INTERCOM V3.0
DATE   04/02/75
TIME   14.38.02.
LOGIN, CABWWALLIS, 1188880321, SUP.
COMMAND-COMRADE, SHARP

COMRADE    TIME:     14.39.55.
           DATE:     04/02/75

SHARP SUBSYSTEM ENTERED
????UPDATE


YOU HAVE SELECTED THE SHARP UPDATE OPTION.

YOU MAY ENTER NEW TRANSACTIONS AND/OR UPDATE THE MASTER FILE.

SELECT ONE OF THE FOLLOWING TRANSACTION FILES:
1   SDD
2   9PTSTB
3   NLCCA
4   YURSO
5   NLCCB
??4

WOULD YOU LIKE TO ENTER ANY NEW TRANSACTIONS? - YES

WILL YOUR INPUT BE ON PAPER TAPE? - NO
PLEASE ENTER YOUR UPDATE TRANSACTIONS.

ID YURSO
A  SSBN 609  1) SAM HOUSTON  3)C  4) 7201  5) 7503  6) 33624
     7) 867  8) 34509  9) 33624  10) 0  11) 867  12) 34509  13) COR
     14) 1  $END

C  SSN 671  7) 1234B  $END

C  AE 27  4) 6401  $END
D  SSN 673  $END
D  DDG 17  15)X  $STOP
```

Figure 8-3 - On Line File Maintenance (Computer Output is Underlined)

shown in Figure 8-2. The user then indicates he wishes to enter new transactions and that his transactions will not be entered on paper tape. He is then instructed to enter his update transactions. Following this, he types in the file ID followed by the add, changes and delete transactions he wishes to enter for this file. The transactions are entered in the format described in Section 4. The user enters one add, two changes and two delete transactions.

Figure 8-4 illustrates the interactive diagnostics output by the system following the validity checking of the input transactions. In this case, element 7 was found to be in error in the change transaction for record SSN 671 because the user typed an alphabetic character "B" into a field defined as integer (see element 7, Fig. 8-2). Element 4 in the second change transaction was also found in error because the first two characters were not in the range 65 to 80 as shown in the data definition.

The user is then asked if he wishes to update the file with the current transactions. The NO response indicates that the transactions will be recorded onto a "holding" file. The user may then initiate an update later by returning to the terminal and answering YES to the same question. Each time the user executes the UPDATE command, he may initiate an update from current holding file, enter new transactions to be merged with holding file and then update, or enter transactions to be merged with holding file but not update. In this case, the user logs off the terminal following inputting the transactions. The user could have re-entered UPDATE at this point and correctly re-entered the two data elements which failed the validity checking.

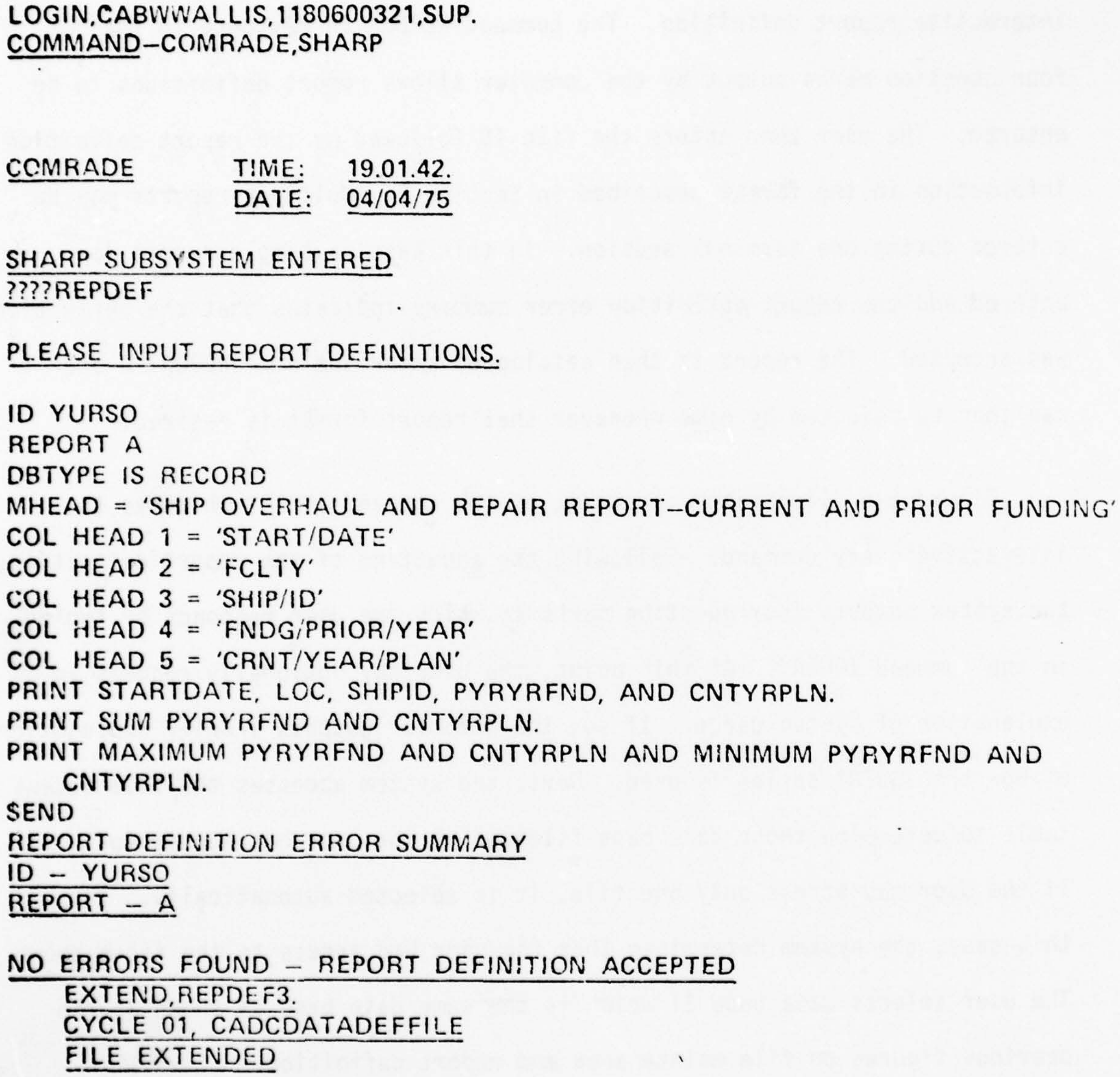
```
THE FOLLOWING FIELDS WERE FOUND TO BE IN ERROR AND IGNORED

RECORD:  SSN 671

     ELEMENT:  007 ERROR:   INTEGER/REAL ERROR - FIELD(S) OR

     SUBFIELD(S) IN ERROR:  1234B


RECORD:  AE 27

     ELEMENT:  004 ERROR:  SUBFIELD RANGE FIELD(S) OR

     SUBFIELD(S) IN ERROR:  64


EXTEND, OUTDISC

CYCLE 01, CADCSD0054

FILE EXTENDED

WOULD YOU LIKE TO UPDATE THE MASTER FILE USING THE
CURRENT TRANSACTIONS? - NO

????LOGOUT.
```

Figure 8-4 - Validity Check Diagnostics

Figure 8-5 illustrates the usage of the command REPDEF for on line interactive report definition. The command REPDEF in response to the four question marks output by the computer allows report definitions to be entered. The user then enters the file ID followed by the report definition information in the format described in Section 6. Multiple reports may be entered during one terminal session. In this case, a single report is entered and the report definition error summary indicates that the definition was accepted. The report is then catalogued under the name REPORT A and can then be selected by name whenever that report format is desired.

Figure 8-6 illustrates the usage of the command IQUERY which is the interactive query command. Following the inputting of the report definition, the system outputs four question marks to which the user responds by typing in the command IQUERY. At this point, the user may optionally request an explanation of system usage. If so, the computer outputs a brief explanation of how the IQUERY option is used. Next, the system accesses the file access table to determine those data base files which the user has access to. If the user may access only one file, it is selected automatically. In this case, the system determined that the user had access to the files shown. The user selects data base 31 which is the same data base as shown in the previous figures on file maintenance and report definition.

The user is next told to select either the advanced or tutorial query mode. The advanced query mode allows the usage of the English like query language described in Section 5. The tutorial query mode is a step by step



```
LOGIN,CABWWALLIS,1180600321,SUP.
COMMAND-COMRADE,SHARP


COMRADE         TIME:   19.01.42.
                DATE:   04/04/75

SHARP SUBSYSTEM ENTERED
????REPDEF

PLEASE INPUT REPORT DEFINITIONS.

ID YURSO
REPORT A
DBTYPE IS RECORD
MHEAD = 'SHIP OVERHAUL AND REPAIR REPORT-CURRENT AND PRIOR FUNDING'
COL HEAD 1 = 'START/DATE'
COL HEAD 2 = 'FCLTY'
COL HEAD 3 = 'SHIP/ID'
COL HEAD 4 = 'FNDG/PRIOR/YEAR'
COL HEAD 5 = 'CRNT/YEAR/PLAN'
PRINT STARTDATE, LOC, SHIPID, PYRYRFND, AND CNTYRPLN.
PRINT SUM PYRYRFND AND CNTYRPLN.
PRINT MAXIMUM PYRYRFND AND CNTYRPLN AND MINIMUM PYRYRFND AND
    CNTYRPLN.
$END
REPORT DEFINITION ERROR SUMMARY
ID - YURSO
REPORT - A

NO ERRORS FOUND - REPORT DEFINITION ACCEPTED
    EXTEND,REPDEF3.
    CYCLE 01, CADCDATADEFFILE
    FILE EXTENDED
```

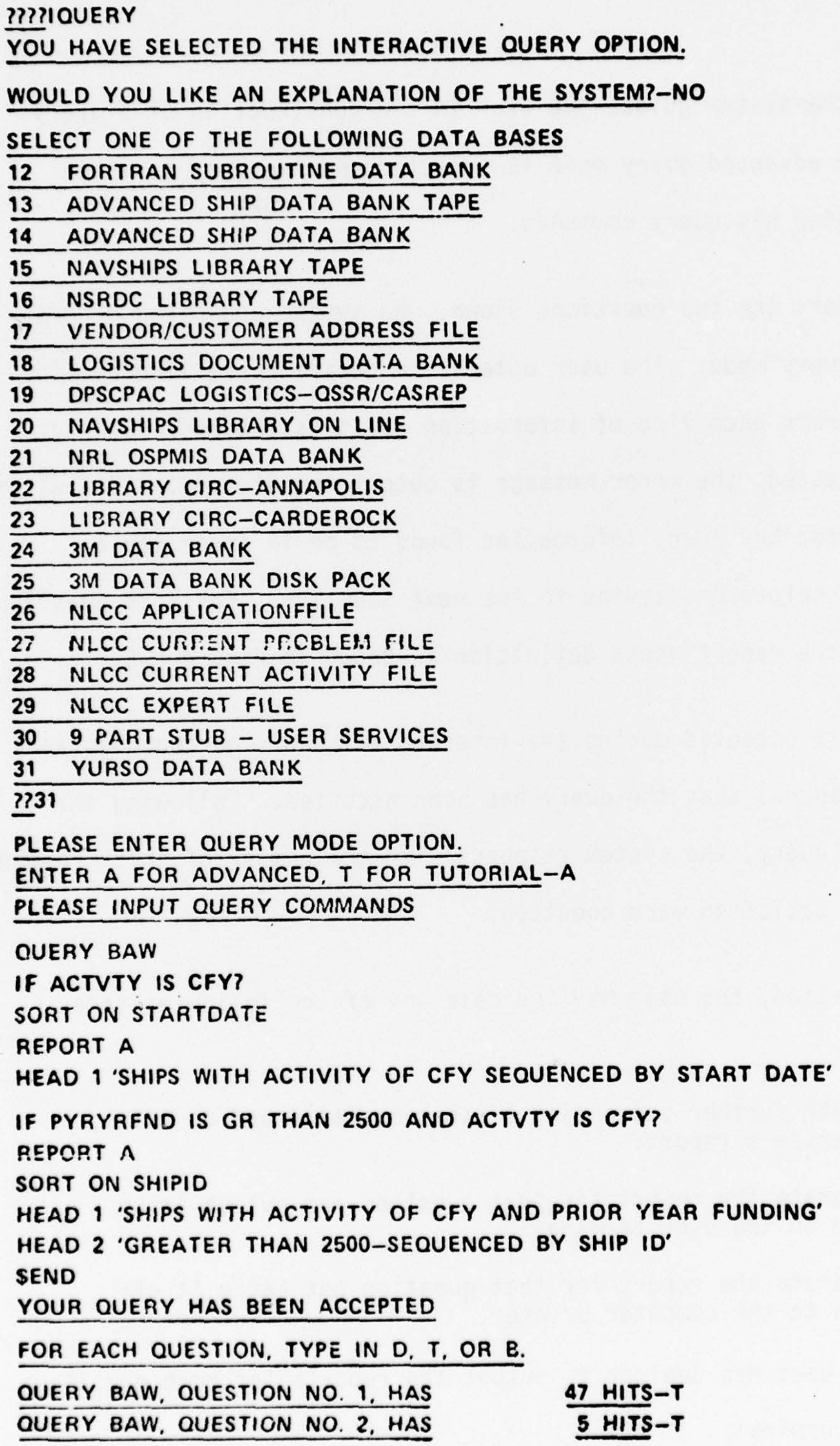
Figure 8-5 - Interactive Report Definition

AD-A055 581   DAVID W TAYLOR NAVAL SHIP RESEARCH AND DEVELOPMENT CE--ETC   F/G 9/2
THE SHARP DATA BASE MANAGEMENT SYSTEM - USER INFORMATION MANUAL--ETC(U)
AUG 76   B WALLIS, R HYDE, T BLAKE

UNCLASSIFIED   DTNSRDC-TM-188-76-1   NL

3 OF 3
AD
A055 581

END
DATE
FILMED
8-78
DDC



```
????IQUERY
YOU HAVE SELECTED THE INTERACTIVE QUERY OPTION.

WOULD YOU LIKE AN EXPLANATION OF THE SYSTEM?-NO

SELECT ONE OF THE FOLLOWING DATA BASES
12  FORTRAN SUBROUTINE DATA BANK
13  ADVANCED SHIP DATA BANK TAPE
14  ADVANCED SHIP DATA BANK
15  NAVSHIPS LIBRARY TAPE
16  NSRDC LIBRARY TAPE
17  VENDOR/CUSTOMER ADDRESS FILE
18  LOGISTICS DOCUMENT DATA BANK
19  DPSCPAC LOGISTICS-QSSR/CASREP
20  NAVSHIPS LIBRARY ON LINE
21  NRL OSPMIS DATA BANK
22  LIBRARY CIRC-ANNAPOLIS
23  LIBRARY CIRC-CARDEROCK
24  3M DATA BANK
25  3M DATA BANK DISK PACK
26  NLCC APPLICATIONFFILE
27  NLCC CURRENT PROBLEM FILE
28  NLCC CURRENT ACTIVITY FILE
29  NLCC EXPERT FILE
30  9 PART STUB - USER SERVICES
31  YURSO DATA BANK
??31

PLEASE ENTER QUERY MODE OPTION.
ENTER A FOR ADVANCED, T FOR TUTORIAL-A
PLEASE INPUT QUERY COMMANDS

QUERY BAW
IF ACTVTY IS CFY?
SORT ON STARTDATE
REPORT A
HEAD 1 'SHIPS WITH ACTIVITY OF CFY SEQUENCED BY START DATE'

IF PYRYRFND IS GR THAN 2500 AND ACTVTY IS CFY?
REPORT A
SORT ON SHIPID
HEAD 1 'SHIPS WITH ACTIVITY OF CFY AND PRIOR YEAR FUNDING'
HEAD 2 'GREATER THAN 2500-SEQUENCED BY SHIP ID'
$END
YOUR QUERY HAS BEEN ACCEPTED

FOR EACH QUESTION, TYPE IN D, T, OR B.

QUERY BAW, QUESTION NO. 1, HAS            47 HITS-T
QUERY BAW, QUESTION NO. 2, HAS             5 HITS-T
```

Figure 8-6 - Interactive Query

procedure where the system guides the user in the construction of a query. In this case, the advanced query mode is selected and the user is given the message to enter his query commands.

The user enters the two questions shown. No system prompting occurs in the advanced query mode. The user enters the entire query line by line. The system interprets each line of information as it is entered. Should any errors be detected, the error message is output immediately. This allows the user to re-enter any query information found to be in error for a specific question before proceeding to the next question. For both questions, the user selects the report whose definition is shown in Figure 8-5.

No errors were detected during the interpretation of the two questions and the system responds that the query has been accepted. Following the resolution of the query, the system responds with the number of "hit" records in the file which satisfied each question.

For each question, the user may indicate any of the following three options -

D - Delete further processing of that question and do not generate a report.

T - Generate the report for that question and output it on line on the user terminal.

B - Generate the report for that question but batch it off line to the computer printer.

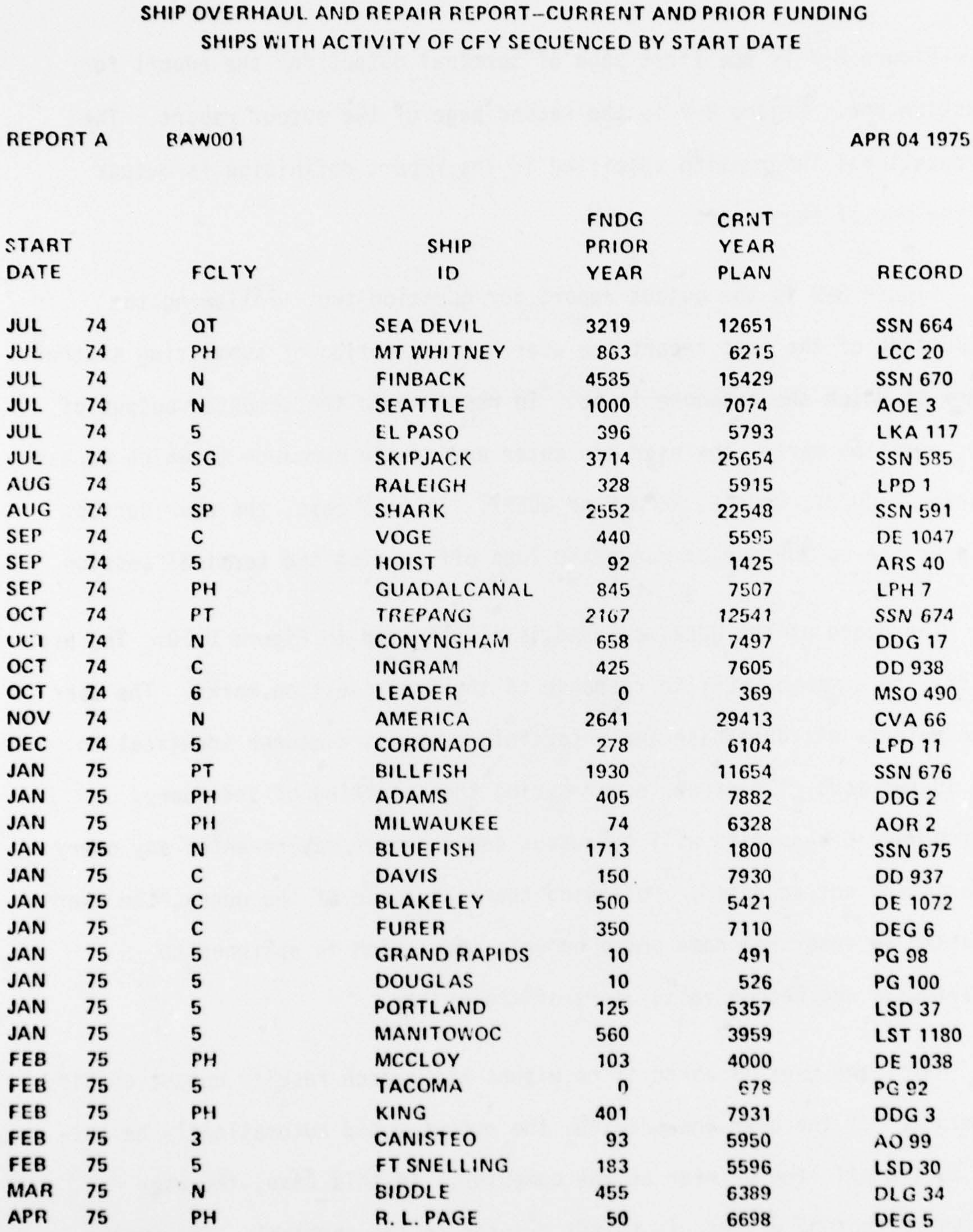
In this case, the user has decided to output the reports for both questions onto the on line terminal.

Figure 8-7 is the first page of terminal output for the report for question one. Figure 8-8 is the second page of the output report. The computational information specified in the report definition is output at the end of the report.

Figure 8-9 is the output report for question two. Following the outputting of the last report the user has the option of submitting another query to which the response is no. In response to the computer output of four question marks, the user can enter any of the commands to which he has access - IQUERY, UPDATE, REPDEF or QUERY. In this case, the user decides to exercise no further commands and logs off, ending the terminal session.

The usage of the QUERY command is illustrated in Figure 8-10. The user enters the command QUERY in response to the four question marks. The user then selects his data base and enters his query in a manner identical to the IQUERY mode. If errors occur during the inputting of the query, appropriate diagnostics will be output and the user may re-enter any query information not accepted. Following the acceptance of the query, the user is asked to enter his name and a numeric code which is assigned to individuals who become valid users of the system.

Next, the user is asked if he wishes his search results output on his terminal. If the user answered NO, the output would automatically be output to the off line printer at the computer. In this case, the user indicates that he wishes his output printed on the terminal. The system then indicates that the query run will be executed in the batch mode. The user then logs off and awaits the execution of the batch run.

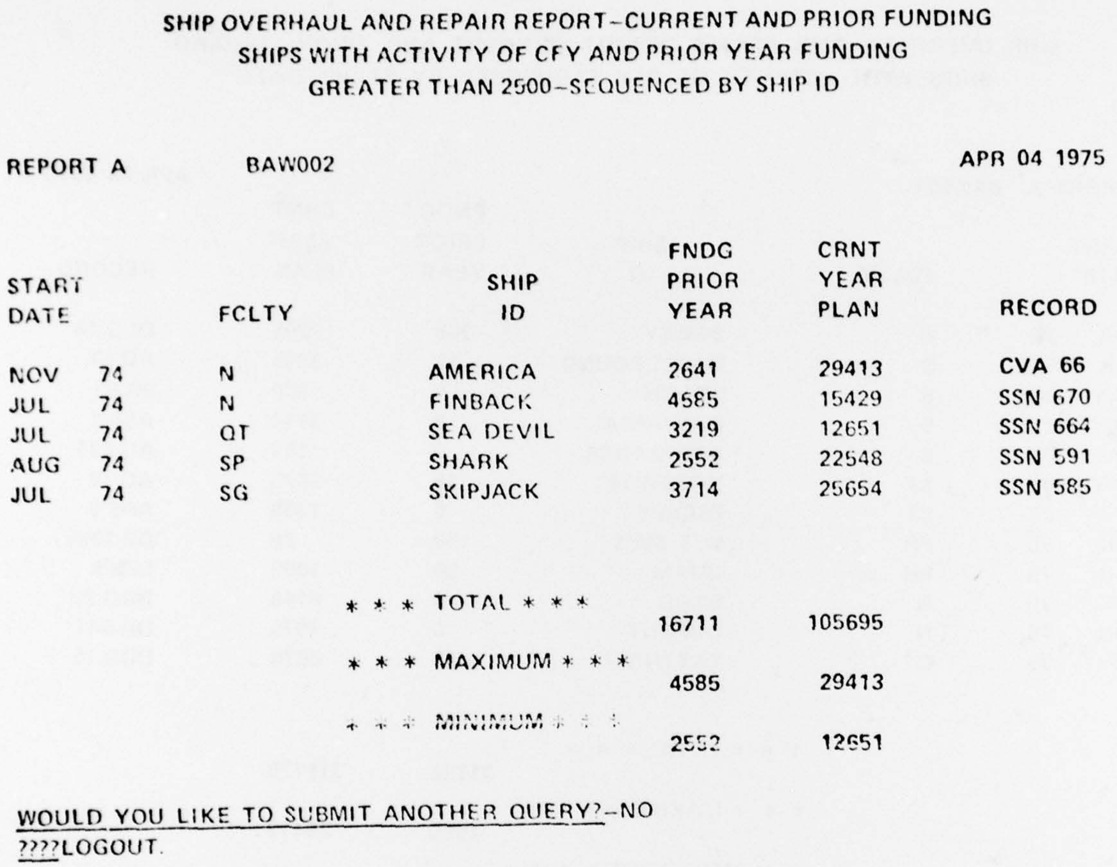


```
          SHIP OVERHAUL AND REPAIR REPORT—CURRENT AND PRIOR FUNDING
               SHIPS WITH ACTIVITY OF CFY SEQUENCED BY START DATE

REPORT A      RAW001                                                  APR 04 1975

                                                 FNDG      CRNT
START                           SHIP             PRIOR     YEAR
DATE          FCLTY             ID               YEAR      PLAN        RECORD

JUL    74     QT         SEA DEVIL               3219     12651        SSN 664
JUL    74     PH         MT WHITNEY               863      6215        LCC 20
JUL    74     N          FINBACK                 4535     15429        SSN 670
JUL    74     N          SEATTLE                 1000      7074        AOE 3
JUL    74     5          EL PASO                  396      5793        LKA 117
JUL    74     SG         SKIPJACK                3714     25654        SSN 585
AUG    74     5          RALEIGH                  328      5915        LPD 1
AUG    74     SP         SHARK                   2552     22548        SSN 591
SEP    74     C          VOGE                     440      5595        DE 1047
SEP    74     5          HOIST                     92      1425        ARS 40
SEP    74     PH         GUADALCANAL              845      7507        LPH 7
OCT    74     PT         TREPANG                 2167     12541        SSN 674
OCT    74     N          CONYNGHAM                658      7497        DDG 17
OCT    74     C          INGRAM                   425      7605        DD 938
OCT    74     6          LEADER                     0       369        MSO 490
NOV    74     N          AMERICA                 2641     29413        CVA 66
DEC    74     5          CORONADO                 278      6104        LPD 11
JAN    75     PT         BILLFISH                1930     11654        SSN 676
JAN    75     PH         ADAMS                    405      7882        DDG 2
JAN    75     PH         MILWAUKEE                 74      6328        AOR 2
JAN    75     N          BLUEFISH                1713      1800        SSN 675
JAN    75     C          DAVIS                    150      7930        DD 937
JAN    75     C          BLAKELEY                 500      5421        DE 1072
JAN    75     C          FURER                    350      7110        DEG 6
JAN    75     5          GRAND RAPIDS              10       491        PG 98
JAN    75     5          DOUGLAS                   10       526        PG 100
JAN    75     5          PORTLAND                 125      5357        LSD 37
JAN    75     5          MANITOWOC                560      3959        LST 1180
FEB    75     PH         MCCLOY                   103      4000        DE 1038
FEB    75     5          TACOMA                     0       678        PG 92
FEB    75     PH         KING                     401      7931        DDG 3
FEB    75     3          CANISTEO                  93      5950        AO 99
FEB    75     5          FT SNELLING              183      5596        LSD 30
MAR    75     N          BIDDLE                   455      6389        DLG 34
APR    75     PH         R. L. PAGE                50      6698        DEG 5
```

Figure 8-7 - Output Report Question 1

```
        SHIP OVERHAUL AND REPAIR REPORT-CURRENT AND PRIOR FUNDING
             SHIPS WITH ACTIVITY OF CFY SEQUENCED BY START DATE


REPORT A  BAW001                                                 APR 04 1975
                                           FNDG      CRNT
START                        SHIP          PRIOR     YEAR
DATE          FCLTY           ID           YEAR      PLAN        RECORD

APR    75     C          DEWEY              205      7205        DLG 14
APR    75     5          PUGET SOUND         15      3335        AD 38
MAY    75     5          WELCH                0       679        PG 93
MAY    75     5          SYLVANIA             0      3714        AFS 2
MAY    75     6          ASSURANCE            0       353        AG 521
MAY    75     SJ         YOSEMITE            15      2675        AD 19
MAY    75     SJ         ESCAPE               0      1358        ARS 6
JUN    75     PH         W. S. SIMS         150        26        DE 1059
JUN    75     PH         GUAM                30      1000        LPH 9
JUN    75     N          BYRD                 5      8146        DDG 23
JUN    75     N          DUPONT               0      4975        DD 941
JUN    75     C          TATTNALL            63      8674        DDG 19


                    * * * TOTAL * * *
                                          31798    317175
                   * * * MAXIMUM * * *
                                           4585     29413
                   * * * MINIMUM * * *
                                              0        26
```

Figure 8-8 - Output Report Question 1 (Cont.)



```
         SHIP OVERHAUL AND REPAIR REPORT-CURRENT AND PRIOR FUNDING
              SHIPS WITH ACTIVITY OF CFY AND PRIOR YEAR FUNDING
                   GREATER THAN 2500-SEQUENCED BY SHIP ID

REPORT A          BAW002                                          APR 04 1975

                                            FNDG       CRNT
START                           SHIP        PRIOR      YEAR
DATE            FCLTY            ID         YEAR       PLAN        RECORD

NOV    74       N            AMERICA        2641       29413       CVA 66
JUL    74       N            FINBACK        4585       15429       SSN 670
JUL    74       QT           SEA DEVIL      3219       12651       SSN 664
AUG    74       SP           SHARK          2552       22548       SSN 591
JUL    74       SG           SKIPJACK       3714       25654       SSN 585


                      * * *  TOTAL * * *
                                           16711      105695

                      * * *  MAXIMUM * * *
                                            4585       29413

                      * * *  MINIMUM * * *
                                            2552       12651

WOULD YOU LIKE TO SUBMIT ANOTHER QUERY?-NO
????LOGOUT.
```

Figure 8-9 - Output Report Question 2

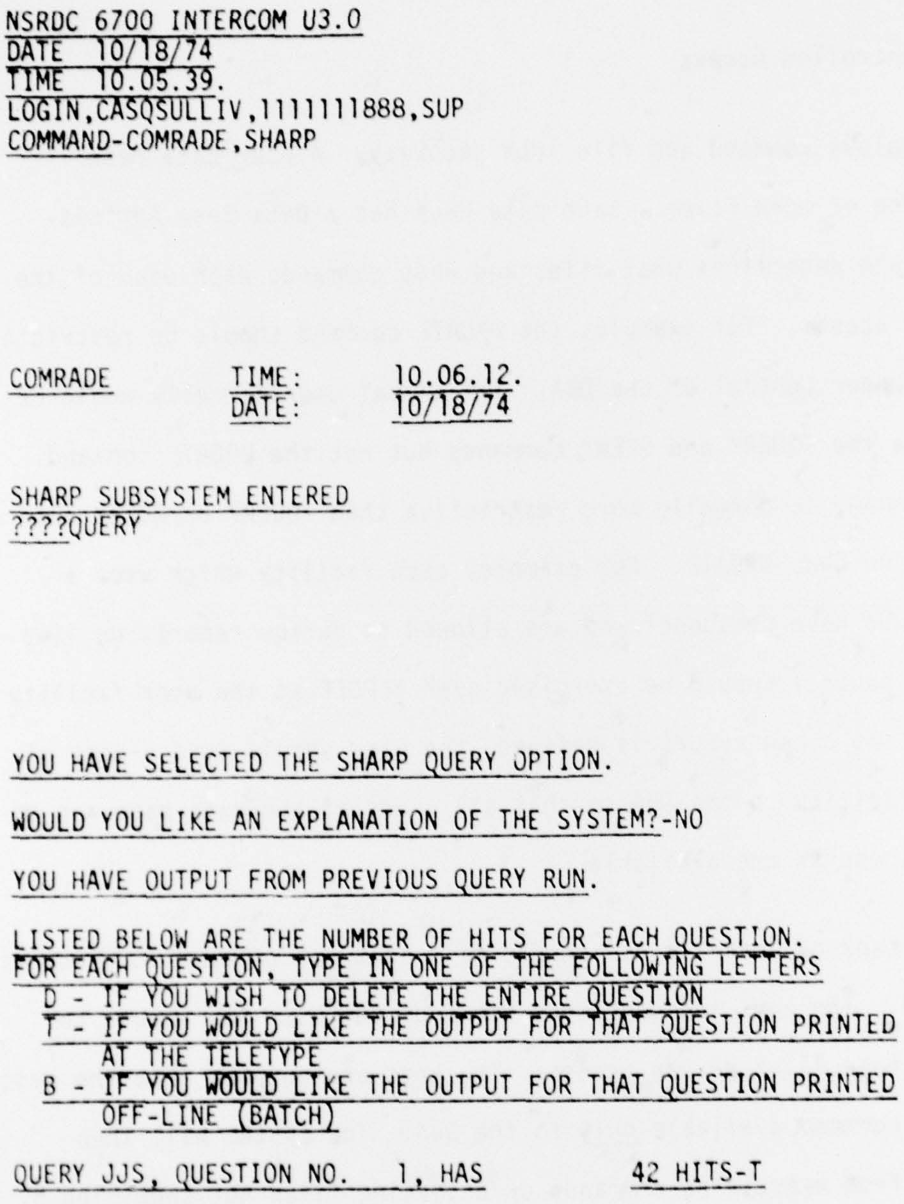
```
NSRDC  6700  INTERCOM  U3.0
DATE  10/18/74
TIME  10.04.50
LOGIN, CASQSULLIV, 1111111888, SUP.
COMMAND - COMRADE, SHARP

COMRADE     TIME:     11.59.10
            DATE:     11/07/74

SHARP SUBSYSTEM ENTERED
???? QUERY

YOU HAVE SELECTED THE SHARP QUERY OPTION.
WOULD YOU LIKE AN EXPLANATION OF THE SYSTEM? - NO

DO YOU WISH TO SUBMIT A QUERY RUN? - YES

SELECT ONE OF THE FOLLOWING DATA BASES

24  3M DATA BANK - F14 SAMPLE
25  3M DATA BANK - F14 ONE MONTH
??  25

PLEASE INPUT QUERY COMMANDS

QUERY JJS
IF WUC IS 65431?   REPORT A21
$END
YOUR QUERY HAS BEEN ACCEPTED

ENTER NAME, CODE FOR PROPER OUTPUT DISPOSAL
NAME - CARDER
CODE - 1882

WOULD YOU LIKE OUTPUT BACK AT TELETYPE? - YES

YOUR QUERY WILL NOW BE SUBMITTED AS A BATCH JOB

???? LOGOUT
```

Figure 8-10 - Batch Terminal Example

Figure 8-11 illustrates how a user may return to a terminal and retrieve the results of the batch query run submitted earlier shown in Figure 8-10. The user logs on and enters the command QUERY. The system then checks and determines that the query run submitted has executed and search results are available for output, if desired. The system then outputs the number of hit records for the question. The user is given the same three options available in the interactive mode - delete the output report, output the report on line, or output the report to the off line printer. In this case the user specifies that the report be output on the terminal. The search results report is then output on the terminal (not shown in the figure).

The batch terminal (QUERY) command is used for low priority queries or when a large number of hit records are anticipated. The user should anticipate a 30 minute to one hour wait from the time the query is input to the time he may return to the terminal to obtain the query results.

A specific user may have only <u>one</u> outstanding QUERY run at a time. For example, if a user submits a QUERY run, he cannot submit another query run until he has disposed of the results of the first run. Note that a user returns to the terminal for output results only if he answers "YES" to the question "WOULD YOU LIKE OUTPUT BACK AT TELETYPE?" (See Figure 8-10). Otherwise, the output is automatically batched off line.



```
NSRDC 6700 INTERCOM U3.0
DATE  10/18/74
TIME  10.05.39.
LOGIN,CASQSULLIV,1111111888,SUP
COMMAND-COMRADE,SHARP



COMRADE          TIME:        10.06.12.
                 DATE:        10/18/74

SHARP SUBSYSTEM ENTERED
????QUERY



YOU HAVE SELECTED THE SHARP QUERY OPTION.

WOULD YOU LIKE AN EXPLANATION OF THE SYSTEM?-NO

YOU HAVE OUTPUT FROM PREVIOUS QUERY RUN.

LISTED BELOW ARE THE NUMBER OF HITS FOR EACH QUESTION.
FOR EACH QUESTION, TYPE IN ONE OF THE FOLLOWING LETTERS
  D - IF YOU WISH TO DELETE THE ENTIRE QUESTION
  T - IF YOU WOULD LIKE THE OUTPUT FOR THAT QUESTION PRINTED
      AT THE TELETYPE
  B - IF YOU WOULD LIKE THE OUTPUT FOR THAT QUESTION PRINTED
      OFF-LINE (BATCH)

QUERY JJS, QUESTION NO.   1, HAS          42 HITS-T
```

Figure 8-11 - Batch Terminal Example (Cont.)

8.3 SHARP Controlled Access

SHARP employs command and file lock security. A user data base consists of one or more files. Each data base has a Data Base Administrator (DBA) who determines what files and what commands each user of the data base may access. For example, the UPDATE command should be restricted to personnel under control of the DBA. The casual user normally would be allowed to use the IQUERY and QUERY commands but not the UPDATE command. The REPDEF command is normally more restrictive than IQUERY or QUERY but less restrictive than UPDATE. For example, each facility which uses a data base should have personnel who are allowed to define reports on line. However, some control should be exercised over REPDEF at the user facility level. Each time a new report is defined, the user should send a copy of the report definition to the DBA so that all users of the data base can be informed what reports are available.

Each new user of SHARP is assigned a file lock and command lock access key by the DBA. The keys indicate what commands the user has access to and what data base files he can access. The keys are entered into the system via a special command available only to the DBA. The system will then prevent users from exercising commands or accessing files not specified by their access keys.

9. SHARP ERROR MESSAGES

9.1 Query Error Messages

QUERY XXX NOT ENTERED

Query ID not entered preparatory to entering query. (e.g., QUERY BST)

IMPROPER SEARCH CRITERION

One of the valid search criteria listed in Figure 5-2 was not entered in the question.

SEARCH VALUE NOT OF FORM A/B

The search values were not entered in the form A/B for a ranging search criteria (BE, BL, BN, BH).

PARTIAL FORM INCORRECT

The partial search value was not one of the forms listed in Section 5.1.4.10.

LINK TAG > 1 CHARACTER

The link tag entered with the search value was greater than one character. It should be a one character alphabetic.

IMPROPER LINK/ROLE ENTRY

Improper syntax was used in specifying a link or role value with a search value.

ROLE VALUE > 2 CHARS

More than two characters were entered for a role value.

INVALID SEARCH ACRONYM

The search acronym entered was not found in the data definition of the file being queried.

SEARCH VALUE NOT ALPHANUMERIC

The data element in the query requires an alphanumeric search value.

SEARCH VALUE NOT NUMERIC

The data element in the query requires a numeric search value (all characters are numbers).

SEARCH VALUE NOT ALPHABETIC

The data element in the query requires an alphabetic search value.

SEARCH VALUE NOT INTEGER

The data element in the query requires an integer search value.

SEARCH VALUE NOT REAL

The data element in the query requires a real search value.

SEARCH VALUE TOO LARGE

The search value was larger than the maximum size specified in the data definition.

SEARCH VALUE WRONG SIZE

The search value was not the exact size specified in the data definition.

FATAL ERROR (Followed by any of the above messages)
QUERY XXX, QUESTION NN
ABOVE QUESTION WAS NOT PROCESSED. PLEASE EXAMINE
FOR SYNTAX ERROR AND CONTENT, RE-ENTER
IF POSSIBLE

This message indicates that for query ID XXX and question number NN, the fatal error indicated was detected and the question was rejected. The user may re-enter the question correctly. The message is output only during interactive query processing.

```
FATAL ERROR IN SELECT, PRINT OR SORT
STATEMENT, QUERY XXX, QUESTION NN
THE TERM  AAAAAAAAAA WAS NOT FOUND IN
USER TERM TABLE.  THAT TERM AND ANY FOLLOWING
TERMS IN THE STATEMENT WERE NOT PROCESSED.
RE-ENTER CORRECT TERM AND ANY FOLLOWING TERMS
IN A SEPARATE STATEMENT.
```

This message indicates that for the question cited an improper acronym was entered in a select, print or sort statement. The following example illustrates. (Assume that DATE is misspelled)

```
PRINT AUTHOR, DOTE AND TITLE
```

The above error message would be output indicating that DOTE was incorrect. To re-enter date correctly, the user would then enter -

```
PRINT DATE AND TITLE
```

Note that only the acronym in question and any following acronyms in the statement need be re-entered.

```
FATAL ERROR IN SELECT PRINT OR SORT
STATEMENT, QUERY  XXX, QUESTION NN
ILLEGAL PREFIX OR PARTIAL SYNTAX IN ABOVE
STATEMENT.  THE INCORRECT TERM AND ALL
FOLLOWING TERMS NOT PROCESSED.  RE-ENTER
CORRECT TERM AND ANY FOLLOWING TERMS IN A
SEPARATE STATEMENT.
```

Example:

```
SORT ON TITLE AND (   ) SOURCE.
```

Error message is output indicating incorrect prefix (no number in parentheses). The user would then re-enter:

```
SORT ON (6) SOURCE.
```

9.2 Report Definition Error Messages

FATAL ERRORS FOUND - REPORT DEFINITION REJECTED.

Indicates that one or more fatal errors were found which did not allow the report to be defined. The specific fatal errors would then be defined as shown below.

(FATAL) KEYWORD EXPECTED BUT NOT FOUND.

Indicates an expected keyword not found such as LINE, PRINT, REPORT, ID, etc.

(FATAL) NO FILE NAME FOLLOWING KEYWORD ID.

Indicates that a file name was not entered following "ID"

(FATAL) FILE NAME FOLLOWING KEYWORD ID DOES NOT EXIST

Indicates file name entered does not exist.

(FATAL) KEYWORD REPORT NOT FOLLOWED BY VALID REPORT NAME

Report ID entered does not exist for file selected.

(FATAL) KEYWORDS LINE AND PRINT BOTH REQUESTED FOR SAME REPORT.

Either but not both LINE and PRINT may be used in the same report definition.

(WARNING) KEYWORD N/M NOT PRECEDED BY DATA NAME

Indicates either N or M not preceded by element acronym name. Report not rejected, N/M is ignored.

(WARNING) NO VALID DATA NAMES FOLLOWING KEYWORD LINE OR PRINT

Indicates that LINE or PRINT was entered followed by another keyword without data element names being specified. Condition can be corrected by re-entering LINE or PRINT statement correctly.

(FATAL) ELEMENT NNNNNNNNNN DOES NOT EXIST ON SPECIFIED FILE

Indicates an invalid (probably misspelled) element acronym was entered.

(FATAL) MORE THAN SIXTY ELEMENTS REQUESTED FOR A PRINT SPECIFICATION

A print statement in a row report causes one element to be output per line. A maximum of 60 lines may be defined for a row report. Thus, no more than 60 elements may be entered in a print statement.

(FATAL) MORE THAN 100 ELEMENTS DEFINED IN A ROW REPORT

A maximum of 100 data elements may be defined in a single row report definition.

(FATAL) INPUT ACRONYM EXCEEDS 10 CHARACTERS

Data element acronyms are limited to 10 characters.

(FATAL) NO DOLLAR SIGN OR QUOTE PRECEDES HEADING

Headings are bounded by dollar sign or single quote marks.

(FATAL) NO HEADING FOLLOWS KEY WORD MHEAD

(FATAL) NO DOLLAR SIGN OR QUOTE FOLLOWS HEADING

(FATAL) KEYWORD REPORT NOT ENCOUNTERED

Report ID must follow file ID.

(FATAL) KEYWORD MHEAD NOT ENCOUNTERED

Each report must have a master heading.

(FATAL) NO ELEMENTS REQUESTED FOR REPORT

At least one element must be specified.

(FATAL) KEYWORD ID MISSING

(FATAL) HEADING LINE FOR A COLUMN EXCEEDS 12 CHARACTERS

(FATAL) NO $ OR QUOTE PRECEDES COLUMN HEADING INFORMATION

(FATAL) NUMBER OF COLUMN HEADING LINES EXCEEDS THREE

(FATAL) A COMPUTATION ON ELEMENT NOT INTEGER OR REAL

A computation such as total or average was specified for a non computational element.

9.3 Retrieval Error Messages

THE FOLLOWING SEARCH VALUE NOT FOUND IN THE USER'S FILE - NNN....N

Indicates that search value N was not located in an "equal" type search. User should check to ensure that search value was spelled correctly.